VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

IMPLEMENTACE PROTOKOLU SIP V PBX ASTERISK
SIP IMPLEMENTATIONS IN ASTERISK OPEN SOURCE PBX

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Vít Bednář**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Pavel Šilhavý, Ph.D.**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Vít Bednář ***ID:*** 155140
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Implementace protokolu SIP v PBX Asterisk

POKYNY PRO VYPRACOVÁNÍ:

Porovnejte a popište nativní implementaci protokolu SIP a PJSIP v PBX Asterisk. Podrobně se věnujte funkcionalitě dílčích modulů PJSIP stacku. Pro obě implementace vytvořte komentované konfigurace pro jednotlivé scénáře uplatnění ústředny. S využitím testeru realizujte testování a vyhodnoťte přínos nově nasazované knihovny PJSIP. Připravte komentované šablony obvyklých scénářů uplatnění v rozsahu dle dohody s vedoucím práce. Vytvořte zadání laboratorní úlohy seznamující s vlastnostmi a konfigurací PJSIP v PBX Asterisk.

DOPORUČENÁ LITERATURA:

[1] Bosse, J.G.. Signaling in telecommunication networks. John Wiley & Sons, Ltd. En-gland 2002 , ISBN 0-47--66288-7.

[2] Collins, D. Carrier grade voice over IP / 2nd ed. New York : McGraw-Hill, 2003. ISBN 0-07-140634-4.

[3] Dwivedi, Himanshu. Hacking VoIP :protocols, attacks, and countermeasures / San Francisco : No Starch Press, 2009. ISBN 978-1-59327-163-3.

[4] Endler, David. Hacking exposed VoIP :voice over IP security secrets & solutions / New York : McGraw-Hill, 2007. ISBN 978-0-07-226364-0.

Termín zadání: 1.2.2017 ***Termín odevzdání:*** 24.5.2017

Vedoucí práce: Ing. Pavel Šilhavý, Ph.D.
Konzultant:

doc. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Práce porovná nativní SIP stack a PJSIP stack v open source ústředně Asterisk. V úvodu je stručně popsán SIP protokol a ústředna Asterisk. Následně je zkoumána architektura, podpora nových funkcí a možnosti nastavení obou stacků. Dále jsou vytvořeny komentované konfigurační soubory pro různé scénáře uplatnění ústředny. Stacky jsou otestovány pomocí zátěžového testeru Spirent TestCenter C1. Následně jsou vyhodnoceny jejich vlastnosti a popsány přínosy nového PJSIP stacku. V závěru je vytvořen návrh laboratorní úlohy

KLÍČOVÁ SLOVA

SIP, PJSIP, ASTERISK, VoIP, Testování

ABSTRACT

The thesis compares native SIP stack with PJSIP stack in the open source telephone private branch exchange (PBX) Asterisk. First, there are described both SIP protocol and Asterisk application. Subsequently, the architecture, new function support and the stacks setting possibilities are explored. For different exchange scenarios several commented configuration files are presented. The stacks are tested using Spirent TestCenter C1 software thereafter. In conclusion, selected properties are assessed and new PJSIP stack benefits are summarized. In addition, the laboratory assignment is attached.

KEYWORDS

SIP, PJSIP, ASTERISK, VoIP, Testing

BEDNÁŘ, Vít *Implementace protokolu SIP v PBX Asterisk*: diplomová práce. BRNO: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2017. 80 s. Vedoucí práce byl Ing. Pavel Šilhavý , , Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Implementace protokolu SIP v PBX Asterisk“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

BRNO ....

podpis autora(-ky)

PODĚKOVÁNÍ

Velmi děkuji vedoucímu diplomové práce panu Ing. Pavlovi Šilhavému, Ph.D., za účinnou metodickou, pedagogickou a odbornou pomoc a podnětné návrhy k práci a také za jeho trpělivost a ochotu při zpracování této diplomové práce.

BRNO ....

podpis autora(-ky)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

V posledních letech minulého století zaznamenali nárůst telefonní ústředny 5. generace využívající pro svoji funkci počítačové sítě založené na protokolu IP, tento typ telefonie se nazývá Voice over IP telefonie zkráceně VoIP. Výhodou těchto sítí je ekonomická nenáročnost, protože pro svoji funkci využívají zpravidla již vybudované počítačové sítě. K signalizaci se ve VoIP sítích využívá zejména protokolu SIP. Tento protokol zaznamenává od svého vzniku znatelný vývoj a je tak kladen důraz na ústředny, aby dokázali implementovat vylepšení a nové funkce tohoto protokolu.

Cílem této práce je tak porovnat implementaci protokolu SIP v open source pobočkové ústředně ASTERISK. O implementaci protokolu v ústředně se starají tzv. kanálové ovladače neboli stacky. V této práci je porovnáván nativní stack, který vznikl již v roce 2002 a nově nasazovaný PJSIP stack založený na knihovnách PJSIP projektu.

V úvodu práce je základní seznámení s protokolem SIP a open source pobočkovou ústřednou Asterisk. Dále je popsán nový PJSIP stack, jeho architektura a možnosti konfigurace. Následně jsou oba stacky porovnány dle jejich architektury, rozdílů implementace SIP protokolu a podpory nových funkcí.

V další části jsou vytvořeny scénáře uplatnění ústředny a k nim komentované konfigurace pro oba stacky. V další kapitole jsou oba stacky otestovány na maximální počet aktivních relací a transakcí za sekundu. Na základě těchto testů jsou vyhodnoceny rozdíly stacků. V poslední části je vytvořeno zadání laboratorní úlohy do předmětu Telekomunikační a informační systémy sloužící studentům k seznámení s PJSIP stackem.

1 VOIP TELEFONIE

Technologie VoIP Voice over IP (Internet Protocols) umožňuje přenášet multimediální data přes IP sítě s využitím sady protokolů. Mezi tyto protokoly patří signalizační protokoly a protokoly pro přenos multimediálních dat. Signalizační protokoly zajišťují registraci, vytváření a ukončování hovorů. Nejvíce využívanými signalizačními protokoly jsou SIP (Session Initiation Protocol), H.323, IAX2 (Inter-Asterisk eXchange). Pro přenos multimediálních dat se z pravidla využívá protokol RTP (Real Time Protocol) či jeho zabezpečená verze SRTP. Na transportní vrstvě je zpravidla využíván protokol UDP. Pro připojení do PSTN sítě se využívá tzv. Getaway (bran), které zajistí převod signalizace i konverzi dat.[3]

Důležitou součástí VoIP telefonie jsou pobočkové ústředny, které se starají o registraci klientů a zprostředkování hovorů. Ústředen existuje celá řada. Jedná se buď o proprietární řešení firem, jako jsou CISCO, NOKIA (Alcatel-Lucent) nebo Panasonic. Další část pak tvoří ústředny s otevřeným kódem tzv. open source ústředny. Mezi největší hráče patří Asterisk, Freeswitch a PBX YATE. Poslední součástí VOIP telefonie jsou koncová zařízení. Ty může představovat hardwarový telefon, softwarový klient v PC nebo aplikace v chytrém telefonu.[1]

1.1 Signalizační protokol SIP

SIP (Session Initiation Protocol) signalizační textově orientovaný klient – server protokol sloužící k navázání, modifikaci a ukončení spojení mezi dvěma či více účastníky. První návrh standartu vznikl v roce 1999 v RFC 2543. V roce 2002 vznikl nový standart RFC 3261. V dnešní době existují stovky další RFC, které se buď SIPu přímo týkají nebo na něj navazují. Pro samotný přenos multimediálních dat se zpravidla používá protokol RTP (Real-Time Protocol), pro vyjednávání parametrů například typ přenášených dat se využívá protokol SDP (Session Description Protocol).[4]

Pro funkčnost protokolu SIP jsou definovány dvě základní SIP entity:

- **SIP User agent (UA)** – představuje koncový bod signalizace
- **SIP server** – obsahuje SIP proxy, registrar, redirect server

Zpravidla bývají všechny SIP servery logickými entitami na jednom HW zařízení.

1.1.1 Sip entity

SIP User agent (UA)

User agent je koncový bod signalizace v SIP sítích. UA může být hardwarový telefon, softwarový klient, B2BUA, PSTN brána atd. Úkolem UA je vytvořit multimediální

tok. User agent client (UAC) generuje žádosti a přijímá odpovědi, user agent server (UAS) přijímá žádosti a generuje odpovědi. UAC a UAS jsou pouze logické entity a každé koncové zařízení má v sobě obě tyto části.

SIP Proxy Server

Úkolem SIP proxy serveru je přesměrovávat žádosti od UA blíže k volanému, zpravidla na další proxy server. Proxy Server nikdy negeneruje SIP žádosti, pouze je přeposílá k volanému.

Registrar server

Server přijímá zprávy SIP REGISTER od UA, tedy slouží k registraci a aktualizuje lokalizační databázi.

Redirect server

Slouží k přesměrování. Přijímá požadavky na vyhledání v lokalizační databázi a zasílá odpověď s lokalizací uživatele zpět tazateli.[5]

1.1.2 SIP zprávy

Komunikace mezi UA a SIP Servery je zajišťována pomocí SIP zpráv. Ty se dělí na SIP ŽÁDOSTI, někdy označovány jako SIP METODY, a SIP ODPOVĚDI. Sip zprávy obsahují hlavičku a tělo zprávy. Na prvním řádku hlavičky je uveden typ zprávy, hlavička je oddělena volným řádkem od těla zprávy. Níže je uvedena hlavička sip žádosti INVITE:

```
Request-Line:INVITE sip:200@192.168.20.23;transport=UDP SIP/2.0
Via:SIP/2.0/UDP192.168.20.120:5060;branch=z9hG4bK-524287-1—3abf760e20a8cc74
Max-Forwards: 70
Contact: <sip:100@192.168.20.120:5060;transport=UDP>
To: <sip:200@192.168.20.23;transport=UDP>
From: <sip:100@192.168.20.23;transport=UDP>;tag=817a2873
Call-ID: ZA1hDnv44N7u6vpTR1QwDg..
CSeq: 2 INVITE
Allow: INVITE, ACK, CANCEL, BYE, NOTIFY, REFER, MESSAGE, OPTIONS, INFO, SUBSCRIBE
Content-Type: application/sdp
Supported: replaces, norefersub, extended-refer, timer, outbound, path, X-cisco-serviceuri
User-Agent: Z 3.9.32144 r32121
Allow-Events: presence, kpml
Content-Length:243
```

SIP metody

UAC generuje SIP metody - žádosti, ty žádají po UAS určitou akci. V původním RFC 3261 bylo definováno šest základních žádostí.

- **INVITE** – žádost o spojení nebo modifikaci již existujícího spojení
- **ACK** – konečné potvrzení poslední odpovědi na žádost INVITE
- **BYE** – žádost o ukončení spojení
- **REGISTER** – žádost o registraci nebo odregistraci
- **CANCEL** – žádost o ukončení během sestavování spojení
- **OPTIONS** – zjištění vlastností protistrany

S postupným rozvojem VoIP byli uvedeny další SIP metody, které přináší nové funkce.

- **PRACK**– plní stejnou funkci jako ACK s tím rozdílem, že potvrzuje dočasné odpovědi. Tím se zajistí spolehlivé doručení.
- **UPDATE** – umožňuje změnit parametry relace (kodeky streamu) bez nutnosti čekat na dokončení metody INVITE a zasláním re-INVITE.
- **REFER** – žádost o předání hovoru. Poslání metody REFER zajistí, že příjemce bude kontaktovat další UA s požadavkem na sestavení spojení.
- **MESSAGE** – zasílání krátkých zpráv
- **SUBSCRIBE, NOTIFY** – přihlášení a zasílání zpráv o změně stavů vzdáleného uzlu (node)
- **INFO**– přenos signalizačních informací během hovoru. [1]

SIP odpovědi

Na každou metodu musí být odpovězeno. Výjimku tvoří pouze metoda ACK, která potvrzuje doručení poslední odpovědi na žádost INVITE. Odpovědi se dělí do 6-ti skupin a jsou značeny tří-číselným kódem:[4]

- **1xx** – dočasné informativní odpovědi (na žádosti, které byly přijaty, ale výsledek ještě není znám) např. 100 (Trying)
- **2xx** – kladná finální odpověď (200 OK)
- **3xx** – přesměrování
- **4xx** – negativní odpověď 401 (Unauthorized)
- **5xx** – chyba na straně serveru 503 (Service Unavaiable)
- **6xx** – globální chyba

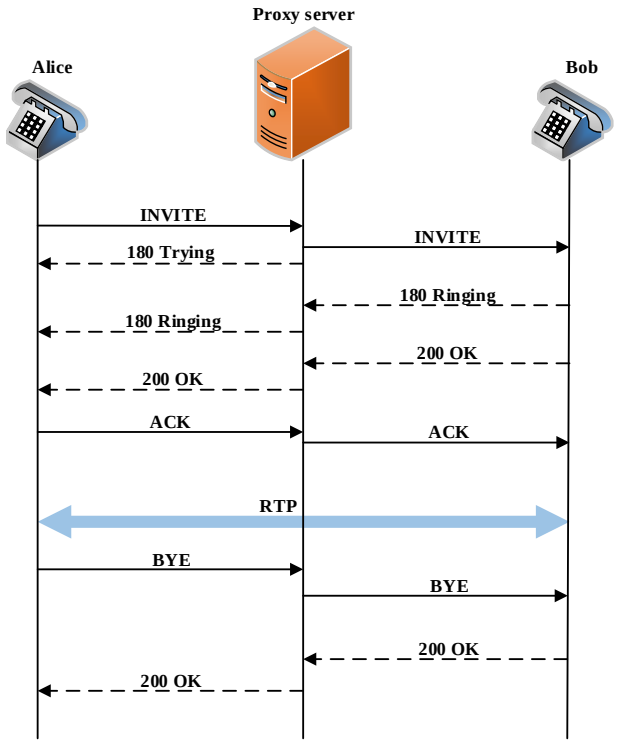
1.1.3 Komunikace v SIP

V nejjednodušší konfiguraci je možné použít dvě SIP entity, pomocí SIP zprávy komunikují mezi sebou, zpravidla však bývá v SIP síti přítomna ještě ústředna. Ústředna zpravidla pracuje v proxy režimu, kdy přeposílá sip žádosti blíže ke klientovi a multimediální data prochází přímo mezi koncovými entitami. Speciálním

případem komunikace je tzv. B2BUA. Kdy jsou vytvořena dvě spojení vždy mezi klientem a ústřednou a multimediální data procházejí přes ústřednu.

Komunikace pomocí SIP PROXY

Příklad komunikace pomocí proxy serveru je znázorněn na obr. 1.1.

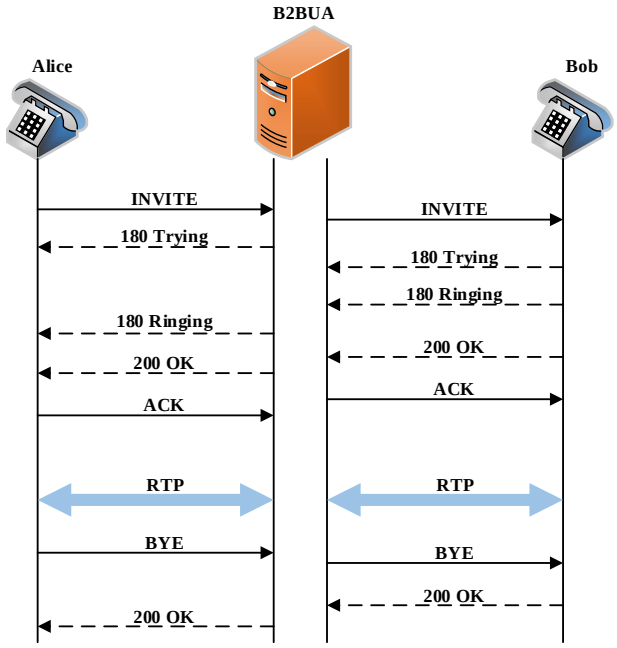


Obr. 1.1: Průběh komunikace s proxy serverem

V doméně spravované jedním SIP Proxy severem se nacházejí klienti Alice a Bob obr. 1.1. Pokud chce Alice kontaktovat Boba, zašle na proxy server žádost INVITE s URI bob@domena.com. Sip server odpoví zprávou 100 Trying a vyhledá v lokalizační databázi IP adresu Boba a přepošle na ni žádost INVITE. Bob potvrdí zprávu odesláním odpovědi 180 Ringing. Pokud je hovor akceptován, je zaslána potvrzující zpráva 200 OK Alici. Alice odpoví též zprávou 200 OK a tím je spojení vytvořeno a mohou být zaslány RTP data. Pokud chce Alice ukončit hovor, zašle zprávu BYE, Bob ukončení potvrdí zprávou 200 OK [6].

Komunikace pomocí B2BUA

Další způsob komunikace je komunikace v pomocí B2BUA, jedná se o speciální případ UA, který je vložen mezi AU. Tento speciální UA je koncovým bodem signalizace. Kde se vytváří jedno spojení mezi klientem Alice a B2BUA a druhé mezi klientem Bob a B2BUA, na rozdíl od Proxy serveru, který komunikaci pouze přeposílá. RTP data tak zpravidla prochází přes B2BUA (ústřednu) a tak má plnou kontrolu nad probíhajícím hovorem. Ovšem výkon ústředny používající tento režim, je menší oproti proxy režimu, díky nutnosti přeposílat RTP data. Toto řešení využívá ústředna Asterisk. Tato ústředna však nabízí možnost zaslat RTP data přímo mezi koncovými entitami[4].



Obr. 1.2: Průběh komunikace pomocí B2BUA

1.2 Open source ústředna Asterisk

Asterisk je dnes nejpoužívanější open source telefonní ústřednou s velkou komunitní základnou. První verzi představil Mark Spencer v roce 1999, poté byla založena firma Digium, která se dále stará o vývoj Asterisku. Hlavní profit firmy plyne z placené technické podpory a prodeje kompatibilního HW. V tab. 1.1 je uveden seznam verzí. Vždy se střídají verze se standardní podporou s verzí, která je dlouhodobě podporovaná LTS (Long Term Support). Během podpory jsou pro obě verze uvolňovány

opravy. Jelikož je zachována stejná architektura a struktura souborů, lze ústřednu snadno upgradovat na novější v dané verzi. U nové verze jsou změny v architektuře a používají se jiné knihovny, např. od verze 12 je přidán nový SIP stack, který je založen na knihovně PJSIP [1][4].

Asterisk podporuje celou řadu režimů:

- VoIP getaway
- Pobočková ústředna (PBX)
- Interaktivní hlasový průvodce IVR
- Konferenční server
- Šifrovací medium
- Překladač čísel
- A další.. [4]

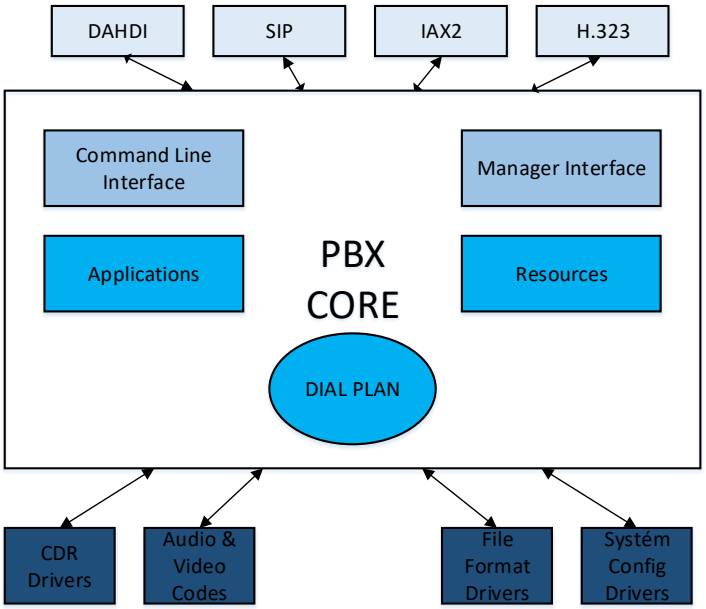
Číslo verze	**Typ verze**	**Datum představení**	**Bezpečnostní opravy**	**Ukončení podpory**
1.2.X		2005-11-21	2007-08-07	2010-11-21
1.4.X	LTS	2006-12-23	2011-04-21	2012-04-21
1.6.0.X	Standard	2008-10-01	2010-05-01	2010-10-01
1.6.1.X	Standard	2009-04-27	2010-05-01	2011-04-27
1.6.2.X	Standard	2009-12-18	2011-04-21	2012-04-21
1.8.X	LTS	2010-10-21	2014-10-21	2015-10-21
10.X	Standard	2011-12-15	2012-12-15	2013-12-15
11.x	LTS	2012-10-25	2016-10-25	2017-10-25
12.x	Standard	2013-12-20	2014-12-20	2015-12-20
13.x	LTS	2014-10-24	2018-10-24	2019-10-24
14.x	Standard	2016-09-26	2017-09-26	2018-09-26

Tab. 1.1: Seznam verzí Asterisku a doba podpory

1.2.1 Struktura Asterisku

Systém Asterisku je navržen, tak aby vytvořil rozhraní mezi telefonní aplikací a telefonním rozhraním (VoIP, TDM). S protokoly SIP, IAX, H323 pracuje Asterisk jako s kanály spojenými s jádrem. Mocným nástrojem pro ovládání Asterisku je příkazový řádek CLI, stejně tak Manager Interface. Základním kamenem je Dialplan, který se stará o přepojování hovorů příchozích i odchozích nebo požadavků na volání služby viz Obr. 1.3 [4].

O správnou funkci každého protokolu se stará tzv. channel driver (kanálový ovladač neboli STACK). V Asterisku je jich několik, název ovladače se pojí s názvem



Obr. 1.3: Architektura Asterisku

protokolu např. chan_sip pro protokol SIP nebo chan_iax2 pro protokol IAX2. Od Asterisku verze 12 je možné využívat dva channel driver pro protokol SIP a to původní chan_sip a nový res_pjsip využívající knihovnu PJSIP. V následujících kapitolách jsou popsány oba stacky.

1.3 Nativní implementace SIP protokolu

Původní ovladač sip kanálu vznikl již v roce 2002. V této době byl SIP protokol poměrně mladý standart, bylo uvolněno RFC 3261 a nebylo zcela zřejmé, jakou pozici si nový protokol vybuduje na poli VoIP. Původní SIP stack byl navržen jako jednokanálový a monolitický s názvem chan_sip.

Konfigurace chan_sip se provádí v textovém souboru sip.conf, ten obsahuje sekci globálního nastavení pro protokol SIP. Tuto sekce uvozuje sekvence [general]. Dále pak názvem či číslem účtu v hranatých závorkách začíná sekce nastavení pro jednotlivé účty např: [100]. Níže je uveden příklad základního. Všechny parametry, které je možné nastavit pro protokol sip nebo jednotlivé účty, jsou uvedeny v literatuře [7].

Obecné nastavení:

```
[general]
context=all                 ;kontext do kterého uživatelé patří
bindaddr=0.0.0.0:5060       ;IP adresa a port na které Asterisk naslouchá
srvlookup=yes               ;povolení DNS
transport=udp               ;povolení protokolu pro přenos
disallow=all                ;zakáže všechny kodeky
allow=alaw                  ;povolení jen kodeku alaw
language=cz                 ;výchozí jazyk
```

Nastavení klienta:

```
[100]                       ;název účtu
type=friend                 ;typ účtů friend
context=kontext1            ;do jakého kontextu klient patří
secret=heslo                ;heslo k účtu
host=dynamic                ;povolení registrace na všech IP adresách
disallow=all                ;zakázání všech kodeků
allow=alaw                  ;povolení jen kodeku alaw
```

2 PJSIP STACK

S vývojem VoIP se postupně vyvíjel i protokol SIP a vznikla řada dalších vylepšení v podobě nových RFC. Úprava původního ovladače chan_sip tak byla stále náročnější a z tohoto důvodu na konferenci AstriDevCon 2012 vývojová komunita rozhodla o implementaci nového stacku. Úkolem bylo nejen navrhnout novou sadu modulů pro protokol SIP pro potřeby dneška, ale byla také snaha, aby byl flexibilní pro úpravy do budoucna. Po analýze různých sip knihoven bylo rozhodnuto o implementaci nového stacku založeného na knihovnách PJSIP, jež sponzoruje společnost Teluu. PJSIP stack je implementován v Asterisku od verze 12 v roce 2014[8].

Důvody vybrání PJSIP:

- Napsaný v C++ což umožnilo snazší imlpementaci do Asteriku
- Je široce používaný, využívá ho řada aplikací, proto je ověřený a mnohokrát testovaný
- Je stále udržován
- Vývojáři již měli s Pjsip zkušenosti z projektu Asterisk SCF

2.1 Funkce podporované PJSIP stackem

PJSIP stack je navržen, aby se více podobal fungování SIP protokolu. Z pohledu běžného uživatele byla snaha zachovat co nejlepší funkčnost shodnou s původním sip stackem. PJSIP stack má však odlišnou architekturu popsanou v kap. 2.2. Konfigurace PJSIP stacku je podstatně komplexnější než u nativního stacku.

Nový stack přináší některé nové funkce po kterých uživatelé původního stacku dlouho volali. Významnou novou funkcí je možnost registrovat více klientů na jeden uživatelský účet. Obsloužit volání na tento účet je možné buď na poslední přihlášený kontakt nebo na všechny nebo kontakty, které jsou k účtu přiřazeny.

Pro usnadnění přechodu na nový stack je možnost využít skript sip_to_res_pjsip.py, který zajistí převod konfigurace z nativního do nového PJSIP stacku. Ovšem skript dobře funguje pouze u nepříliš náročných konfiguracích. Na ústředně je možné pro implementaci SIP protokolu využívat oba stacky současně, s tím rozdílem že každý stack bude naslouchat na jiném portu.

Asterisk verze 12 s PJSIP stackem podporuje tyto funkce:

- Hovory/media
  - Media streamy
  - Caller id
  - DTMF RFC 7433

  - Session Timers
  - PRACK RFC 3262
  - SIP Reason RFC 3326
  - UDP/TCP/TLS WEB socket
  - SDES-SRTP and DTLS-SRTP
  - Digest autentizace
- Předávání hovorů
  - o RFC 3891
  - RFC 3515
- Metoda INFO
  - Přenos XML dokumentů metodou INFO RFC 5168
- Přenos textových zpráv v hovoru
- SIP security freamwork
- Podpora SIP metody OPTIONS
- Změny SIP hlavičky
- T.38 RFC 3362

Od verze 13 byly přidány další funkce:

- Podpora protokolu HEP (Homer Encapsulation Protocol)
- Možnost zaslání stavu zařízení na další entitu (event state compositor)
- Podpora seznamu událostí - RFC 4662

Od verze 14:

- Rozšířené funkcionality DNS – NAPTR/SRV a záznamů AAA
- Možnost zadat počet opakování neúspěšné registrace

Většina těchto funkcí je podporována i v nativním stacku.

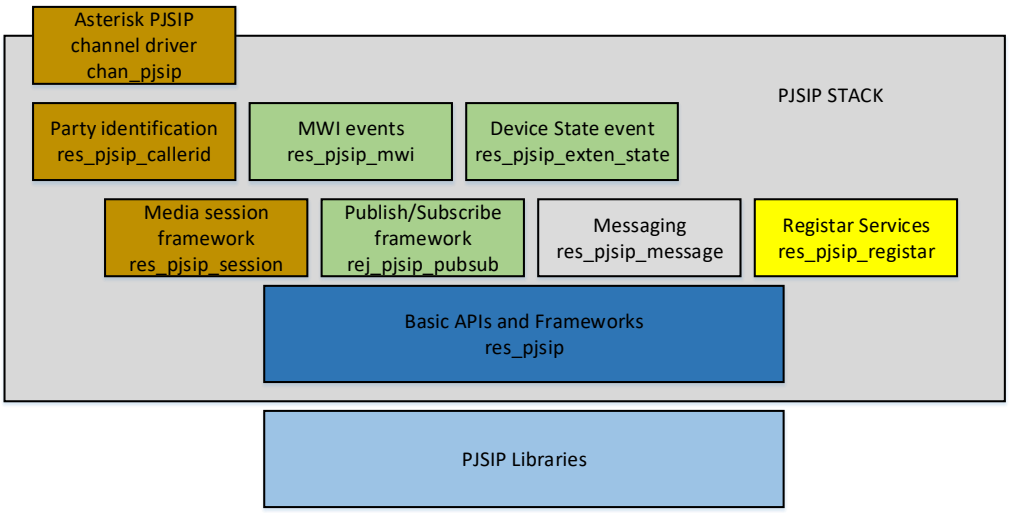
2.2 Struktura PJSIP stacku

Nový sip stack označovaný jako chan_pjsip již není monolitický jednokanálový ovladač, ale vysoce modulární. Stack PJSIP má mnoho malých modulů, z nichž každý poskytuje jinou funkci. Toto modulární řešení přináší řadu výhod:

- Jednotlivé funkce jsou izolované a je tak snadnější tuto funkci upravit či úplně nahradit.
- Přidání nové funkcionality do stacku je snadnější. Například modul res_pjsip_pubsub, který poskytuje metody publish/subscribe, využívají ostatní moduly pro oznámení událostí, například funkce MWI modulu res_pjsip_mwi. To, že

funkce jsou v samostatných modulech má za následek, že v případě přidání nové změny událostí má tato změna jen minimální vliv na stávající systém stacku[8].

Na Obr. 2.1 je uvedena modulární architektura PJSIP stacku. Jsou znázorněny pouze moduly, které zprostředkovávají nejčastěji využívané funkce[4].



Obr. 2.1: Struktura PJSIP stacku

res_pjsip

Tvoří rozhraní mezi stackem a jádrem Asterisku. Je velmi malý oproti původnímu stacku, který měl 34 tis. řádků kódu, má tento jen zhruba 2500 řádků kódu.

res_pjsip_callerid

Tento modul analyzuje údaje ze SIP hlavičky. Zjišťuje informace o ID relace. Vytváří nové ID relace při zahajování hovorů, indetify header pro odchozí zprávy atd.

res_pjsip_session

Při výměně multimediálních dat využívá Asterisk v PJSIP stacku pro popis multimediální relace protokol SDP. Modul res_pjsip_session zajišťuje funkci tohoto protokolu. Zajišťuje také vytváření a ukončení relací.

res_pjsip_pubsub

Modul res_pjsip_pubsub jak název napovídá poskytuje funkce publish/subscribe. Tedy přihlášení se k odběru informací o stavu zařízení a jejich zpracování.

res_pjsip_mwi a res_pjsip_exten_state

Tyto dva moduly úzce souvisí s modulem res_pjsip_pubsub. Modul res_pjsip_mwi má na starosti zpracování MWI zpráv o stavu kanálu na koncových zařízeních. Modul res_pjsip_exten_state poskytuje informace o stavu koncového zařízení.

res_pjsip_messaging

Data potřebná pro vytvoření zpráv generované Asteriskem v PJSIP stacku obstarává modul res_pjsip_messaging. Vyhledává URI podle endpointů. Blokuje nežádoucí zprávy.

res_pjsip_registar

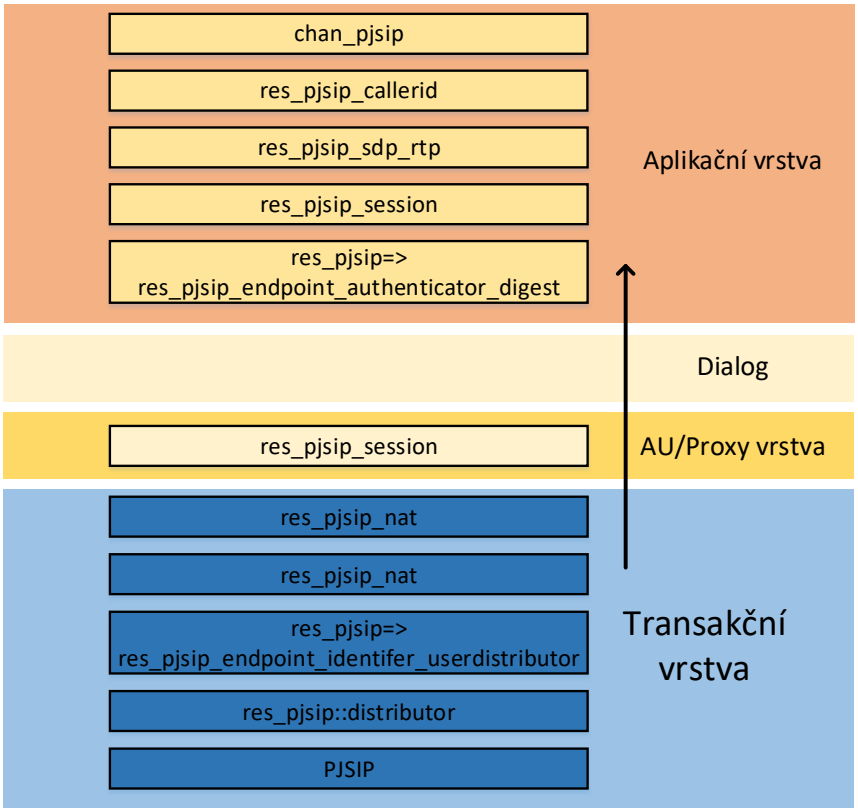
Modul res_pjsip_regisar obstarává příchozí registraci od klientů zpravovaných danou ústřednou, ale i registraci k nadřazené ústředně pokud Asterisk pracuje jako klient. Zajišťuje rovněž vytváření/odebírání kontaktů při registraci a odregistraci.

Detailní popis všech modulů i funkcí, které jednotlivé moduly vykonávají lze nalézt ve zdrojových kódech jednotlivých modulů dostupných na GitHub viz [11].

2.2.1 Zpracování příchozí žádosti INVITE

Na obr. 2.2 je znázorněno, jak je příchozí žádost INVITE zpracována postupně jednotlivými moduly PJSIP stacku od transakční až po aplikační vrstvu. [12]

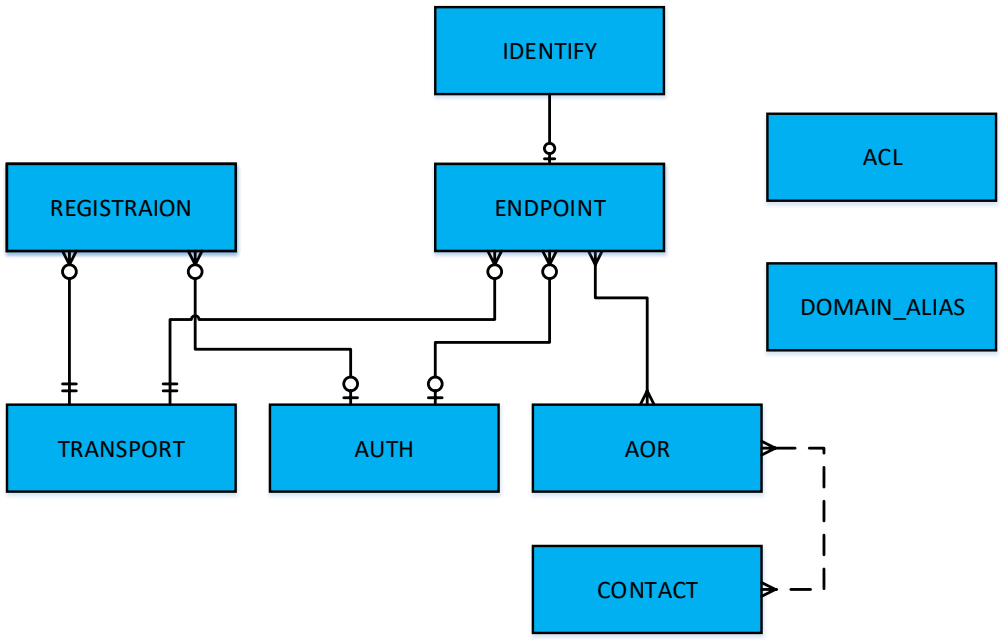
- **res_pjsip::distributor** - předá žádost k dalšímu zpracování
- **res_pjsip_endpoint_identifer** - indeftifikuje endpoint
- **res_pjsip_nat** - upravuje adresy v žádosti
- **res_pjsip_session** - aktualizuje relaci pokud se jedná o re-INVITE
- **res_pjsip_authenticator_digest** - ověřuje žádost
- **res_pjsip_session** - vytváří novou relaci pro každý jednotlivý media stream
- **res_pjsip_sdp_rtp** - zpracovává nabídku medii
- **res_pjsip_caller_id** - získává caller ID a uloží
- **chan_pjsip** - předá zprávu jádru Asterisku [12]



Obr. 2.2: Zpracování žádosti INVITE [12]

2.3 Konfigurace PJSIP stacku

Konfigurace SIP kanálu pomocí nového PJSIP stacku je zcela odlišná od nativního SIP stacku. To je dáno zejména modulární implementací nového stacku, konfigurace se provádí pomocí logických sekcí, kdy určitá logická sekce definuje chování pro určitý modul. Nastavování stacku je tak logičtější, přehlednější a více odpovídá fungování SIP protokolu. Na Obr. 2.3 jsou znázorněny vazby mezi jednotlivými sekcemi.



Obr. 2.3: Vazby mezi sekcemi nastaveni

ENDPOINT

Slouží pro konfiguraci koncového bodu (účtu), např. se zde definuje kontext, používané kodeky a přidružení konkrétních sekcí dalších typů. Jedna sekce typu ENDPOINT může být přidružena k jedné sekci AUTH, TRANSPORT, AOR a INDENTIFY.

TRANSPORT

Určuje chování na transportní vrstvě. Jedna sekce transport může být přidružena k více sekcím ENDPOINT i REGISTRATION.

AUTH

Určuje způsob ověřování při registraci klientů i Asterisku jako klienta. Tato sekce může být přidružena k více ENDPOINT i REGISTRATION.

AOR

Cílem sekce AOR je sdělit Asterisku, kde má kontaktovat ENDPOINT. Bez této sekce nelze ENDPOINT kontaktovat. AOR sekce může být přidružena k více ENDPOINT a CONTACT.

REGISTRATION

Slouží k nastavení pro odchozí registraci (SIP trunk, klient k jiné ústředně). Jedna REGISTRATION sekce může být přidružena k jedné sekci TRANSPORT a AUTH.

DOMAIN_ALIAS

Pokud přichází provoz z domény, pro kterou není v AOR shoda, je možné zvolit pro tuto doménu tzv. alias. Nemá přímý vztah s jinou sekcí.

ACL

Ovlivňuje veškerou příchozí komunikaci přes modul res_pjsip a je nezávislá na koncovém bodu. Nemá přímý vztah s jinou sekcí.

IDENTIFY

Slouží k identifikaci, k jakému koncovému bodu patří příchozí paket. Jedna sekce IDENTIFY může být přidružena k sekci ENDPOINT.

CONTACT

Slouží k vytvoření SIP URI pomocí příchozí registrace nebo ho lze definovat. Je součástí sekce AOR.

2.3.1 Konfigurace pomocí sekcí

Logická sekce nastavení je uvozena v hranatých závorkách:

```
[název_sekce]
volba=hodnota
volba=hodnota
```

Aby bylo možné rozeznat o jakou sekci nastavení se jedná, rozlišují se sekce pomocí položky *type=...* definované uvnitř každé sekce. Například pro sekci typu TRANSPORT je volba type nastavena na *type=transport*.

```
[udp-prenos]          ;název sekce
type=transport        ;typ sekce
```

Název sekce lze ve většině případů volit vlastní. Pokud bude využíván protokol UDP pro přenos, je možné zvolit jako název sekce typu TRANSPORT *[udp-prenos]*. To ulehčí zapamatování funkce sekce. U některých typů sekcí je třeba volit přesný

název sekce, protože to má vliv na její funkci. U sekcí typu ENDPOINT a AOR se musí název shodovat.

Dále je uveden příklad konfigurace sloužících pro základní konfiguraci Asterisku s využitím PJSIP stacku.

ENDPOINT

Tato sekce nabízí základní nastavení funkce SIP pro endpoint. Definuje se zde kontext, povolení či zakázání kodeků, chování na transportní vrstvě přiřazením sekce typu TRANSPORT. Způsob ověřování přiřazením sekce AUTH. A kontaktní údaje díky přiřazení sekce AOR.

```
[6001]                  ;název sekce
type=endpoint           ;typ sekce
context=internal        ;kontext do kterého učet patří
disallow=all            ;zakáže všechny kodeky
allow=alaw              ;povolení kodeku alaw
transport=simpletrans   ;přiřazení sekce transport
auth=auth6001           ;přiřazení sekce AUTH
aors=6001               ;přiřazení sekce AOR
```

TRANSPORT

Mimo protokolu UDP lze využít protokol TCP, šifrovaný TLS nebo WebSocket. PJSIP rovněž podporuje IPv6. Pokud se nachází ústředna za NATem je nutné definovat lokální síť a adresu pro RTP media a SIP signalizaci k ITSP. Je možné vytvořit více sekcí transport, kde například jedna sekce bude sloužit pro hovory v lokální síti a druhá při volání k ITSP.

```
[udp_prenos]                ;název sekce
type=transport              ;typ sekce
protocol=udp                ;protokol pro přenos
bind=0.0.0.0:5060           ;adresa, port na které Asterisk naslouchá
local_net=192.0.2.0/24      ;adresa lokální sítě
external_media_address=     ;adresa pro RTP media, port na které Asterisk na-
                            slouchá
external_signaling_address= ;adresa pro SIP signalizaci
```

AUTH

Pro ověřování uživatele či registraci k ITSP lze kromě ověření pomocí uživatelského jména a hesla lze použít ověření pomocí md5 hash.

```
[auth6001]                  ;název sekce
type=auth                   ;typ sekce
auth_type=userpass          ;typ ověřování
username=6001               ;uživ. jméno
password=6001               ;heslo
```

AOR

AOR (Address of Record) bývá někdy označována jako logická URI uživatele (user URI). U PJSIP stacku je možné, aby se na jeden účet přihlásilo více zařízení např. mobilní telefon a stolní telefon.

```
[6001]                      ;název sekce
type=aor                    ;typ sekce
max_contacts=2              ;maximální počet připojených zařízení
```

REGISTRATION

Sekce registration slouží pro nastavení Asterisku jako klienta připojeného pomocí SIP trunku k ITSP. Tato sekce nastavuje modul res_pjsip_outbo-und_registration.

```
[mytrunk]                   ;název sekce
type=registration           ;typ sekce
transport=simpletrans       ;přiřazení sekce transport
outbound_auth=mytrunk       ;přiřazení sekce auth
server_uri=sip:sip.example.c ;server uri adresa
retry_interval=60           ;opakování neúspěšné registrace
```

DOMAIN_ALIAS

Provoz z domény domena2.com je identifikován jako by přišel z domena.com.

```
[domena.com]                ;název sekce
type=domain_alias           ;typ sekce
domain=domena2.com          ;doména pro kterou se má použít alias
```

ACL

Jedná se o obdobu Access listu. Access list lze definovat jako sekci nebo samostatný soubor acl.conf. Slouží pro nastavení modulu res_pjsip_acl.

```
[acl]                       ;název sekce
type=acl                    ;typ sekce
deny=0.0.0.0/0.0.0.0        ;odmítnuté adresy
permit=209.16.236.0         ;povolení jedné adresy
```

IDENTIFY

Pokud chceme identifikovat provoz z IP 203.0.113.1 jako uživatele 100

```
[100]                       ;název sekce
type=identify               ;typ sekce
endpoint=100                ;jakému klientovi se mají data přiřadit
match=203.0.113.1           ;z jaké IP adresy
```

CONTACT

Slouží k vytvoření SIP URI pomocí příchozí registrace. Kontakty se vytváří automaticky při registraci do AOR nebo je možné vytvořit je manuálně a to položkou contact v sekci AOR.

```
[6001]                              ;název sekce
type=aor                            ;typ sekce
contact=sip:klient@example.com      ;kontakt
```

2.3.2 Využití šablon pro konfiguraci

Konfigurace u nového stacku je poněkud složitější než tomu je u nativního stacku. Pro náročnější konfiguraci například při vytváření většího počtu účtů je možné používat šablony sekcí. Ty výrazně zkrátí a zpřehlední konfiguraci, částečně zamezí chybám v konfiguraci a vytváření redundantního kódu. Šablona se vytvoří přidáním (!) za název sekce, dále se sekce nastavuje obdobně jako v předešlém případě. Pro uplatnění šablony v konfiguraci se za název sekce přidá název šablony v závorkách. Šablony lze obdobným způsobem využívat i v nativním stacku.

Vytvoření šablony sekce typu endpoint a její uplatnění:

```
[endpoint-basic](!)                 ;název šablony
type=endpoint                       ;typ sekce šablony
transport=simpletrans               ;platná sekce transport
context=internal                    ;kontext
allow=ulaw                          ;povolení ulaw
auth=auth1                          ;přiřazení sekce AUTH1
aors=aor1                           ;přiřazení sekce AOR1

[100](endpoint-basic)               ;účet s číslem 100 využívající šablonu
                                    endoint-basic
```

3 POROVNÁNÍ PJSIP S NATIVNÍ IMPLEMENTACÍ

3.1 Architektura

Architektura obou stacků je zcela odlišná, PJSIP stack je na rozdíl od starého ovladače vysoce modulární. Každý modul ve stacku vykonává jinou funkci pro SIP. Modul registar se stará o registraci klientů atd. Tato modulární architektura umožňuje snazší úpravy funkcí a přidávání nových funkcí do stávajících modulů a možnost přidání dalších modulů bez zásahu do jádra stacku. Případná změna modulu má tak minimální vliv na běh celého stacku. Na rozdíl od původní implementace, kde přidání nových funkcí bylo značně složitější.

3.2 Konfigurace

Konfigurace PJSIP stacku je komplexnější než tomu je u nativního stacku. Nastavení se provádí pomocí logických sekcí, kde každá plní jinou funkci pro SIP protokol. V původním stacku jsou v podstatě pouze dvě sekce, kde v jedné lze nastavit obecné vlastnosti pro protokol SIP platné pro všechny účty a v další se nastavují jednotlivé účty, naproti tomu v PJSIP stacku lze nastavit různé vlastnosti SIP pro každý účet zvlášť jak je popsáno v kap. 2.2 a 2.3. Lze tak lépe uzpůsobovat vlastnosti SIP protokolu pro potřeby jednotlivých klientů. Například chování na transportní vrstvě pomocí sekce typu transport či způsob ověřování klientů v sekci auth.

3.3 Registrace více zařízení na jeden účet

PJSIP stack nabízí možnost registrovat více zařízení na jeden endpoint. Je tak možné registrovat stolní telefon, mobilní telefon či softwarového klienta v PC na jeden účet. Maximální počet klientů je definován v sekci AOR položkou max_contacts.

```
[office]            ;název sekce
type=aor            ;typ sekce
max_contacts=3      ;maximální počet připojených klientů
```

Při volání klapky obdobnému v nativním stacku, se stack podívá do sekce AOR a zahájí hovor na první připojený kontakt. Pro volaní na všechny kontakty připojené k danému účtu slouží v příkazu Dial místo parametru PJSIP parametr PJSIP_DIAL_CONTACTS

```
exten => 6001,1,Dial(PJSIP/6001)                ;pouze na první kontakt
```

```
exten => 6001,1,Dial(${PJSIP_DIAL_CONTACTS(6001)}) ;na všechny kontakty
```

3.4 Uživatelské rozhraní CLI

S nasazením PJSIP stacku byla do uživatelského rozhraní CLI implementována i nová sada příkazů obsluhující PJSIP stack. Jelikož je nastavení pomocí nového stacku komplexnější, je například i výpis endpointů obsáhlejší.

Pro výpis informací o konkrétním endpointu např. 100 slouží příkaz *pjsip show endpoint 100*. Ve výpisu na obr. 3.2 je vidět část informací o endpointu a to jaká sekce AOR je používána, kde se má kontaktovat tento endpoint a název používané sekce transport.

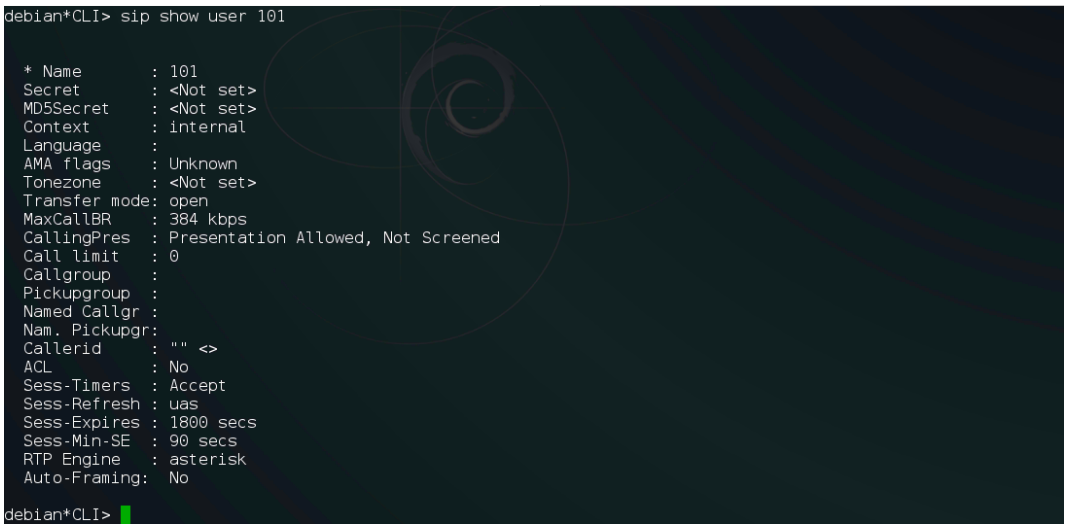
```
debian2*CLI> pjsip show endpoint 400

 Endpoint:  <Endpoint/CID.....................................>  <State.....>  <Channels.>
    I/OAuth:  <AuthId/UserName.....................................................>
        Aor:  <Aor.........................................>  <MaxContact>
      Contact:  <Aor/ContactUri.....................> <Hash....> <Status> <RTT(ms)..>
  Transport:  <TransportId........>  <Type>  <cos>  <tos>  <BindAddress..................>
   Identify:  <Identify/Endpoint.....................................................>
        Match:  <ip/cidr.......................>
    Channel:  <ChannelId.....................................>  <State.....>  <Time.....>
        Exten: <DialedExten...........>  CLCID: <ConnectedLineCID.......>
==========================================================================================

 Endpoint:  400                                           Not in use    0 of inf
    InAuth:  400/400
        Aor:  400                                         2
     Contact:  400/sip:400@192.168.20.120:5060;transport= 525071adcd Unknown        nan
  Transport:  transport-udp            udp      0      0  0.0.0.0:5060
```

Obr. 3.1: Výpis informací o endpointu

Pro porovnání jsou na obr. 3.2 zobrazeny informace o uživateli, které poskytuje nativní stack.

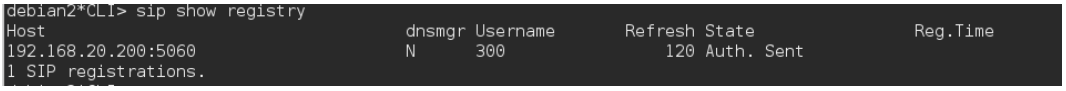


Obr. 3.2: Výpis informací o uživateli

Při výpisu informací o registraci k nadřazené ústředně slouží příkaz *pjsip show registrations*. Z konzole lze vyčíst název trunku, serverovou URI, používanou sekci AOR a stav registrace.

Obr. 3.3: Výpis registrací PJSIP stack

Obdobný výpis poskytuje i nativní sip stack obr. 3.4 Je zde Server URI, uživatelské jméno a stav registrace.



Obr. 3.4: Výpis registrací nativní stack

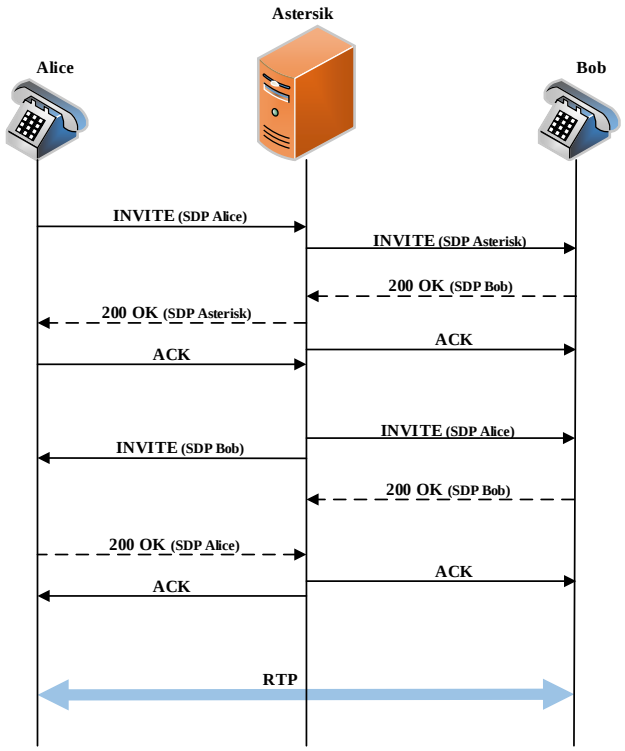
3.5 Rozdíly implentace SIP

3.5.1 Chování v Proxy režimu

Asterisk je navržen jako B2BUA, umožňuje však zasílat multimediální data přímo mezi klienty *tzv. direct media* a tím značně snížit nároky na ústřednu. Pro tuto funkci je nutné splnit několik podmínek:

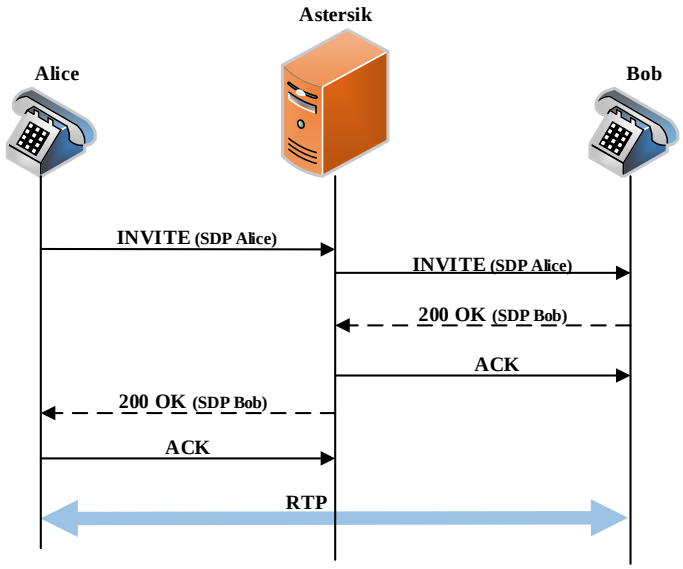
- dva účastníky hovoru
- stejný kodek
- stejné zabezpečení medii (šifrované/nešifrované)
- nelze využít funkce ústředny pracujících s medii (např. nahrávání)

Funkci lze povolit nastavením *directmedia=yes* v sip.conf resp. *direct_media =yes* v pjsip.conf. Tím dojde k tomu, že při sestování hovoru Asterisk zašle volajícímu a volanému nový INVITE, kde v SDP poli se již nenachází adresa a port Asterisku, ale adresa a port volajícího/volaného a multimediální data tak mohou být zasílána přímo mezi volaným a volajícím viz obr. 3.5.



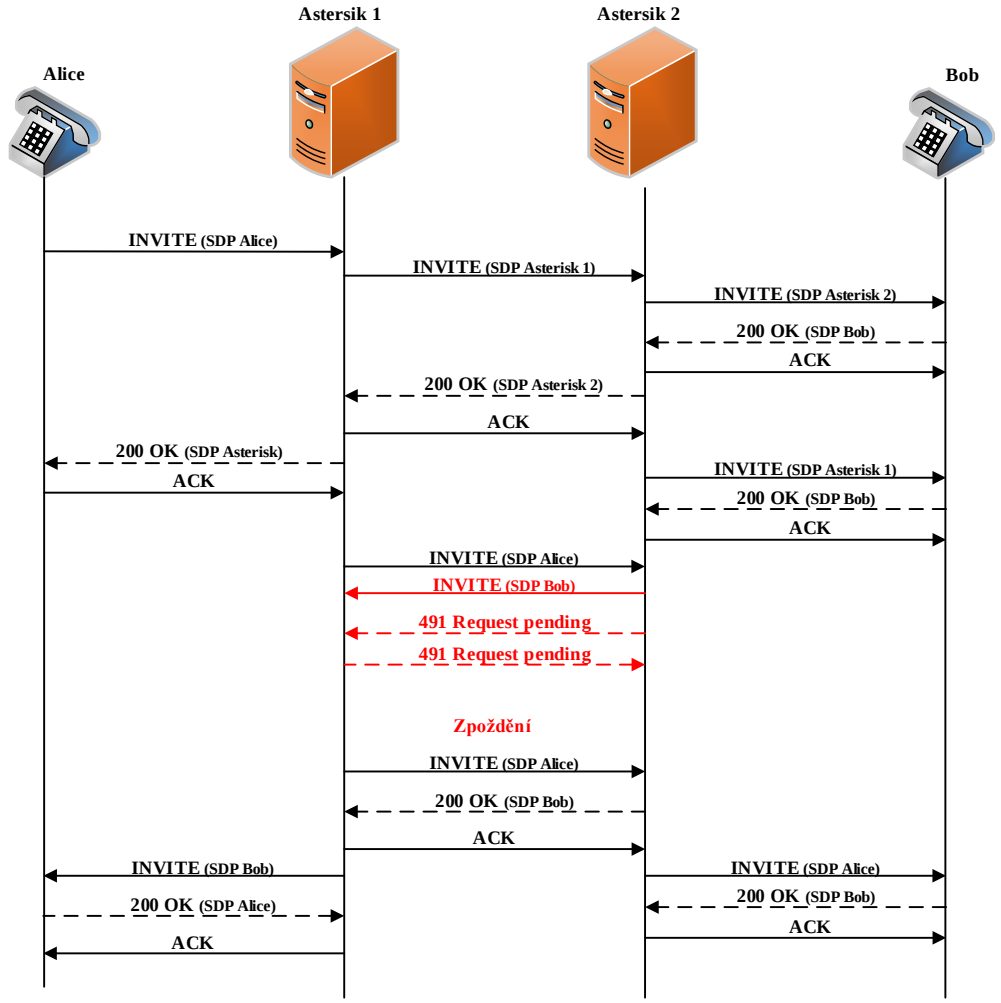
Obr. 3.5: Zprávy při nastaveni direct media

Při tomto nastavení se oba stacky chování obdobně, jiná situace nastane pokud je nastavena v nativním stacku volba *directrtpsetup=yes*. Při zapnutí této volby dochází k přeposlání INVITE metody z volajícího na volaného. V INVITE metodě se v SDP tak již neobjevuje adresa ústředny, ale pouze klientů jak naznačuje obr. 3.6. Tím se zkrátí doba sestavování hovorů a Asterisk se chová téměř jako Proxy server, kdy zprávy INVITE pouze přeposílá a multimediální data jsou zasílána přímo mezi klienty. PJSIP stack však tuto funkci nepodporuje a tak je možné v tomto stacku RTP data zasílat přímo mezi klienty až při obdržení druhé žádosti INVITE.



Obr. 3.6: Zprávy při nastaveni directrtpsetup

Pokud by bylo v cestě více Asterisk serverů, může nastat situace, kdy dojde k zaslání metody RE-INIVTE dvěma servery ve stejný čas a dojde tak ke kolizi, která se projeví chybnou odpovědí *491 Request Pending* obr 3.7. Tento problém lze řešit v nativním stacku nastavením *directmedia=outgoing*. Asterisk tak nebude posílat RE-INIVTE na příchozí hovory. (lze nastavit jak globálně, tak pro určitého klienta). Oproti tomu PJSIP stack nabízí zakázání i na odchozí hovory, zde k tomu slouží volba *direct_media_glare_mitigation=outgoing/incoming*. V obou stacích lze místo INVITE zasílat metodu UPDATE a případně zakázat direct media pokud se nachází klient za NATem (*directmedia=nonat*, *disable_direct_media_on_nat=yes*) [14].



Obr. 3.7: Zaslání RE-INVITE servery současně

3.5.2 Vytváření trunků

Asterisk umožňuje taktéž funkci SIP klienta. Je tak možné připojit Asterisk k nadřazené ústředně pomocí SIP trunku. Pomocí nativního stacku je realizace trunku poměrně jednoduchá. V sip.conf v sekci obecného nastavení [general] přidat položku *registar =>*, která zajistí registraci k nadřazené ústředně.

```
register =>uziv.jmeno:heslo@sipprovider.cz:port
```

Dále je nutné vytvořit další účet typu friend, na který budou směrovány odchozí hovory.

```
[myprovider]                ;název učtu
type=peer                   ;typ účtu
username=uziv.jmeno         ;uživ. jméno
secret=heslo                ;heslo
host=sipprovider.cz         ;adresa providera
canreinvite=no              ;zajistí průchod RTP dat přes ústřednu
insecure=invite             ;ústředna se nautorizuje pro příchozí hovory
context=prichozi            ;kontext
```

V PJSIP stacku je nastavení trunku komplexnější a více tak odpovídá fungování SIP protokolu. Pro vytvoření trunku je nutné nastavit sekce typu REGISTRTION, ENDPOINT, AOR, AUTH a IDENTIFY.

Sekce REGISTRTION zajistí registraci u nadřazené ústředny pomocí uživ. jména a hesla definovaného v sekce ATUH. Sekce ENDPOINT představuje účet, na který budou směrovány odchozí hovory směrem k nadřazené ústředně. Kde se má tato ústředna kontaktovat určuje sekce AOR. Sekce IDENTIFY identifikuje příchozí provoz z IP adresy nadřazené ústředny jako ENDPOINT myprovider.

```
[myprovider]                        ;název
type=endpoint                       ;typ sekce
aors=my-itsp                        ;přiřazení sekce AOR
outbound_auth=myprovider-auth       ;přiřazení sekce AUTH
context=default                     ;kontext

[myprovider]                        ;název
type=aor                            ;typ sekce
contac =sip:sip.my-itsp.net         ;kontaktní adresa
```

```
[myprovider-auth]                          ;název
type=auth                                  ;typ sekce
auth_type=userpass                         ;typ ověřování
username=my_username                       ;uživ. jméno
password=my_password                       ;heslo

[myprovider-reg]                           ;název sekce
type=registration                          ;typ sekce
outbound_auth=myprovider-auth              ;použití sekce auth
server_uri=sip:sipprovider.cz:port         ;URI adresa serveru
client_uri=sip:my_username@sipprovider-    ;klientská URI
.cz:port

[myprovider-identify]                      ;název sekce
type=identify                              ;typ sekce
endpoint=myprovider                        ;přiřazení sekce endpoint
match=sip.myprovider.cz                    ;identifokování podle adresy
```

Pro usnadnění registrace k nadřazené ústředně je v Asterisku od verze 13 možnost využít utility *Configuration Wizard*. Která vyplněním položek v souboru pjsip_wizard.conf zajistí automatické vytvoření všech potřebných sekcí.

```
[myprovider]                               ;název
type=wizard                                ;typ sekce
sends_auth=yes                             ;ověřování
sends_registrations=yes                    ;odeslat rgistraci
remote_hosts=sipprovider.cz:port           ;adresa poskytovatele
outbound_auth/username=my_username         ;uživatelské jméno
outbound_auth/password=my_password         ;heslo
endpoint/context=prichozí                  ;kontext
```

Směrování hovoru na trunk vytvořený PJSIP stackem vypadá následovně:

```
exten =>_100,1,Dial(PJSIP/100@myprovider)
```

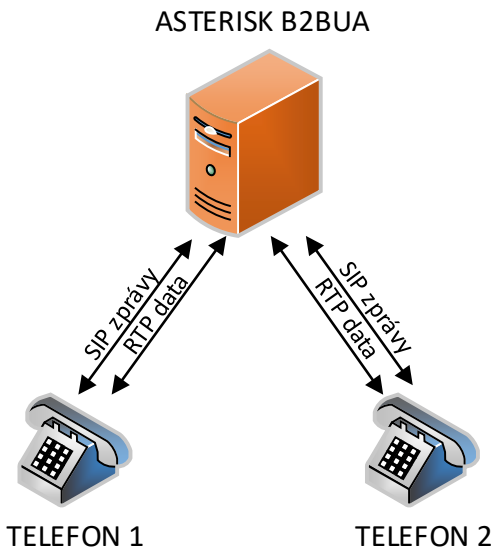
4 SCÉNÁŘE UPLATNĚNÍ PJSIPU A TESTOVÁNÍ STACKŮ

V následující kapitole jsou uvedeny scénáře uplatnění, při kterých bude ústředna pracovat jako B2BUA, proxy server, klient připojený k nadřazené ústředně. V posledním scénáři je realizována zabezpečená komunikace. Pro všechny scénáře je uveden způsob nastavení pomocí nativního stacku a nového PJSIP stacku. V příloze jsou uvedeny konfigurační soubory ústředny pro tyto scénáře. Následně jsou stacky otestovány na maximální počet aktivních relací a transakcí za sekundu.

4.1 Scénáře uplatnění

4.1.1 Režim B2BUA

V této části bude Asterisk pracovat jako jednoduchá ústředna v režimu B2BUA. Jak bylo zmíněno v kap. 1.1.3 v tomto režimu jsou zasílány RTP data přes ústřednu. To sice ústřednu více zatěžuje, ale je zde možnost implementovat funkce nahrávání či změnu kodeku. Schéma funkce ústředny je zobrazeno na Obr. 4.1.



Obr. 4.1: Asterisk v režimu B2BUA

Konfigurace chan_sip

Jak bylo zmíněno v kap 1.3, konfigurace SIP účtů se pomocí starého stacku provádí v konfiguračním souboru sip.conf. Níže je uvedena konfigurace pro režim B2BUA:

Obecná nastavení SIP:

```
[general]
context=all                ;kontext do kterého uživatelé patří
bindaddr=0.0.0.0:5060      ;IP adresa a port na které Asterisk naslouchá
srvlookup=yes              ;povolení DNS
transport=udp              ;povolení protokolu pro přenos
disallow=all               ;zakáže všechny kodeky
allow=alaw                 ;povolení jen kodeku alaw
language=cz                ;výchozí jazyk pro všechny účty
```

Nastaveny jsou pouze základní parametry. Níže je uvedeno nastavení pro jednoho klienta, pro druhého je nastavení obdobné, pouze se změní název účtu:

```
[100]                      ;název účtu
type=friend                ;typ účtů friend
context=sip                ;kontextu klienta
secret=heslo               ;heslo k účtu
host=dynamic               ;povolení registrace na všech IP adresách
```

Nastavení chan_pjsip

Pro konfiguraci pomocí nového stacku je nutné v souboru pjsip.conf nastavit sekce typu *transport, endpoint, auth, aor,* význam těchto sekcí je uveden v kap. 2.3.

```
[100]                      ;číslo klienta
type=endpoint              ;typ sekce
context=pjsip              ;kontext
disallow=all               ;zakázání všech kodeků
allow=alaw                 ;povolení kodeku ulaw
transport=transport-udp    ;platná sekce typu transport
auth=auth100               ;platná sekce typu auth
aors=100                   ;platná sekce typu aors
```

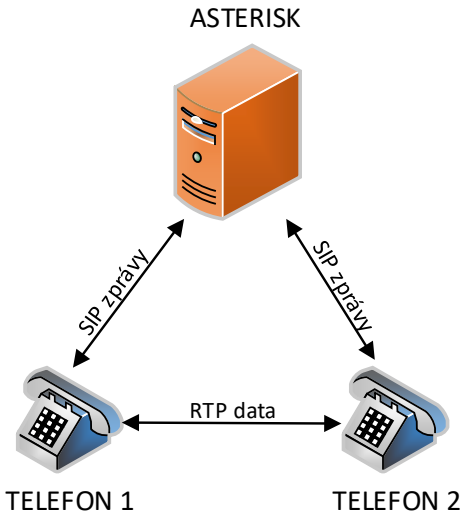
```
[transport-udp]          ;název sekce typu transport
type=transport           ;typ sekce
protocol=udp             ;protokol
bind=0.0.0.0             ;IP adresa na které Asterisk naslouchá

[auth100]                ;název sekce typu Auth
type=auth                ;typ sekce
auth_type=userpass       ;typ autentizace
password=100             ;heslo
username=100             ;uživ jméno

[100]                    ;název sekce
type=aor                 ;typ sekce
max_contacts=1           ;max počet připojených klientů
```

4.1.2 Proxy režim

Asterisk je navržen jako B2BUA, nabízí však možnost přiblížit své chování SIP PROXY serveru. Toho je dosaženo tím, že po sestavení hovoru je znovu zaslána žádost INVITE a RTP data pak mohou být zasílána mezi klienty a nezatěžují tak ústřednu pro ni zbytečným provozem viz kap. 3.5.1. Schéma funkce ústředny v tomto režimu je uveden na Obr.4.2.



Obr. 4.2: Proxy režim

Pro tento scénář je nastavení téměř totožné jako v režimu B2BUA, pouze v části obecného nastavení nativního sip stacku je nutno přidat položku *directmedia=yes* respektive *direc_media=yes* v PJSIP stacku, která zajistí zasílaní RTP dat přímo mezi klienty.

V nativním stacku je možné zajistit přeposílání žádosti INVITE na klienta pomocí volby *directrtpsetup* popsané v kap. 3.5.1.

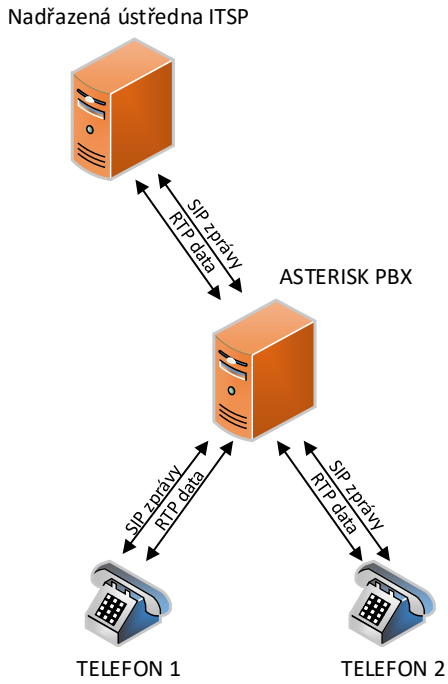
```
[general]
directmedia=yes          ;zaslání RTP dat přímo mezi klienty
context=sip              ;kontext do kterého uživatelé patří
bindaddr=0.0.0.0 :5060   ;IP adresa a port na které Asterisk naslouchá
srvlookup=yes            ;povolení DNS
transport=udp            ;povolení protokolu pro přenos
disallow=all             ;zakáže všechny kodeky
allow=alaw               ;povolení jen kodeku alaw
language=cz              ;výchozí jazyk pro všechny účty
```

V PJSIP stacku je nastavena volba *direc_media=yes*:

```
[100]                          ;číslo klienta
type=endpoint                  ;typ sekce
direct_media=yes               ;zaslání RTP dat přímo mezi klienty
context=pjsip                  ;kontext
disallow=all                   ;zakázání všech kodeků
allow=ulaw                     ;povolení kodeku ulaw
transport=transport-udp        ;platná sekce typu transport
auth=auth100                   ;platná sekce typu auth
aors=100                       ;platná sekce typu aors
```

4.1.3 Klient připojený k nadřazené ústředně pomocí trunku

V následujícím scénáři bude ústředna připojena jako klient k nadřazené ústředně neboli k ITSP pomocí trunku a následně bude plnit funkci pobočkové ústředny, ke které budou připojeni dva klientské telefony. Jak je znázorněno na Obr. 4.3.



Obr. 4.3: Ústředna připojená k ITSP

Konfigurace pomocí nativního stacku, v souboru sip.conf je nejprve nutno se k nadřazené ústředně registrovat, poté je třeba vytvořit účet, na který budou směrovány hovory směřující k nadřazené ústředně. Účet je definován obdobně jako v předchozích případech, je zde ale nutné definovat adresy nadřazené ústředny:

```
[general]
register =>uziv.jmeno:heslo@sipprovider.cz

[myprovider]                ;název učtu
type=peer                   ;typ účtu
username=uziv.jmeno         ;uživ. jméno
secret=heslo                ;heslo
host=sipprovider.cz         ;adresa providera
canreinvite=no              ;zajistí průchod RTP dat přes ústřednu
insecure=invite             ;ústředna se nautorizuje pro příchozí hovory
context=prichozi            ;kontext
```

Jak bylo popsáno v kapitole 3.5.2, k připojení k nadřazené ústředně stačí vyplnit potřebné údaje v pjsip_wizard.conf. A Astersik díky nástroji PJSIP Configuration Wizard zajistí vytvoření všech potřebných sekcí.

```
[mujprovider]                           ;název
type=wizard                             ;typ sekce
sends_auth=yes                          ;ověřování
sends_registrations=yes                 ;odeslat rgistraci
remote_hosts=sipprovider.cz:port        ;adresa poskytovatele
outbound_auth/username=my_username      ;uživatelské jméno
outbound_auth/password=my_password      ;heslo
endpoint/context=prichozí               ;kontext
endpoint/allow=alaw                     ;povoleni kodeku alaw
```

4.1.4 Šifrované spojení

V tomto scénáři bude chování ústředny obdobné jako ve scénáři v kap. 4.1.1 viz obr. 4.1. Ale veškerá komunikace mezi klienty a ústřednou bude šifrována. Pro zabezpečenou komunikaci mezi klientem a ústřednou je nutné zašifrovat jak SIP signalizaci, tak i multimediální data. Pro zašifrování SIP signalizace oba stacky využívají protokol TLS (Transport Layer Security), multimediální data jsou zabezpečena protokolem SRTP. Aby bylo možné využívat v Asterisku SRTP protokol, je nutné mít doinstalované knihovny libsrtp. TLS protokol využívá infrastrukturu veřejných klíčů PKI. Postup vytváření klíčů a certifikátu je uveden například v [10].

Při využívání nativního stacku je v části *[general]* nejprve povolen TLS protokol, následně určena adresa a port pro zabezpečenou komunikaci. Následuje cesta k certifikátu, povolení šifer a nastavení TLS na verzi 1. Dále je nutné u definování klienta určit používání TLS protokolu.

V této fázi je šifrována pouze SIP signalizace, šifrování i multimediálních dat zajistí volba *encryption=yes*.

```
tlsenable=yes                          ;povolení protokolu TLS
tlsbindaddr=0.0.0.0:5061               ;IP adresa a port
tlscertfile=/etc/asterisk/keys/cert.pem ;certifikát
tlscipher=ALL                          ;povolení všech šifer
tlsclientmethod=tlsv1                  ;verze TLS
```

```
[100]
transport=tls                          ;povolení protokolu TLS
encryption=yes                         ;šífrování mutlimediálních dat
...
```

Při využívání PJSIP stacku je nutné definovat či upravit sekci TRANSPORT, aby šifrovala multimediální data. Dále je nutné definovat certifikát, privátní klíč a verzi TLS.

```
[transport-tls]
type=transport                                ;typ sekce
protocol=tls                                  ;protokol TLS
bind=0.0.0.0:5061                             ;IP adresa a port
cert_file=/etc/asterisk/keys/asterisk.crt     ;certifikát
priv_key_file=/etc/asterisk/keys/asterisk.key ;privátnímu klíč
method=tlsv1                                  ;verze TLS
```

Ostatní sekce zůstávají beze změny pouze u sekce typu ENDPOINT je povoleno šifrování multimediálních dat *media_encryption=sdes*. Místo SDES lze využít DTLS-SRTP.

```
[100]
media_encryption=sdes                    ;šifrování multimediálních dat
..
```

Pokud není k dispozici certifikát vytvořený nějakou známou certifikační autoritou CA, je je možné určit certifikační autoritu:

V nativním stacku:

```
[100]
tlscafile=                               ;určení CA
```

U PJSIP stacku v sekci ENDPOINT:

```
dtls_ca_file=                            ;určení CA
```

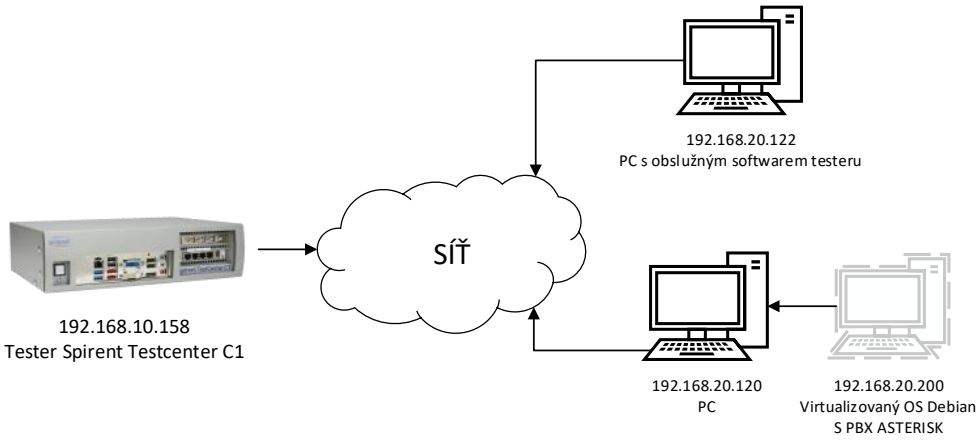
4.2 Specifika testování

Cílem testování stacků je zjistit, zda bude nový pjsip stack díky své modulární architektuře výkonnější než původní stack. Prvním sledovaným výkonnostním parametrem je maximální počet relací. Kdy na ústřednu budou zasílány požadavky na spojení a sledován počet úspěšně spojených hovorů. Druhým výkonnostním parametrem je maximální počet transakcí za sekundu.

Oba parametry budou testovány ve scénářích B2BUA a PROXY. V B2BUA režimu prochází přes ústřednu kromě SIP signalizace i multimediální RTP data, je tak poměrně pravděpodobné, že bude ústředna spíše zatížena přeposíláním RTP mezi klienty.

Ústředna Asterisk je nainstalována na virtualizovaném operačním systému Debian ve verzi 8 Jessie. Debian je open-source operační systém šířený pod licencí GNU/Linux, který vyniká stabilitou a jednoduchou údržbou. K virtualizaci je využívám nástroj VMware Workstation Player verze 12, běžící na základním systému Windows 7 se 4 GB RAM a 4-jádorový CPU Intel core i3 3570 3,4 GHz. Debianu je přiděleno ze systémových prostředků 1,5 GB RAM a jedno jádro procesoru, z důvodu zachování dostatečných systémových prostředků pro systém při přetížení ústředny Asterisk.

Výkonnostní testy budou prováděny na testeru Spirent TestCenter C1. Tester má dvě části serverovou a klientskou, které budou představovat dvě koncové zařízení mezi kterými bude probíhat hovor. Schéma sítě při testování je uvedeno na obr. 4.4.



Obr. 4.4: Schéma sítě při testování

4.3 Instalace Asterisku

Asterisk je možné získat jako samostatnou linuxovou distribuci AsteriskNOW nebo získat zdrojový kód a provést vlastní kompilaci programu. AsteriskNOW je kompletní linuxová distribuce, obsahuje ovladač DAHDI a grafické uživatelské prostřední FreePBX pro správu Asterisku. V této práci byla zvolena možnost instalace Asterisku ze zdroje, z důvodu možnosti vybrat moduly, které budou v Asterisku nainstalovány.

Nejprve byla stažena aktuální verze Asterisku konkrétně verze 14.1 a rozbalen instalační balíček:

```
root@debian:/usr/local/src# wget http://downloads.asterisk.org/pub/telephony/asterisk/asterisk-14-current.tar.gz
root@debian:/usr/local/src# tar -zxvf asterisk-14-current.tar.gz
```

Ke správné funkci nového sip stacku Asterisk vyžaduje knihovny pjproject 12.4 nebo novější. Kompatibilní knihovny lze získat z GitHub nebo jako balíček. Asterisk od verze 13.8 již obsahuje knihovny Pjproject. Pro instalaci knihoven bylo nutné nejprve spustit skript, který nám zajistí instalaci všech potřebných balíčků a knihoven pro kompilaci:

```
contrib/scripts/install_prereq install
```

Příprava na kompilaci s pjsiproject knihovnami a kompilace programu:

```
./configure –with-pjproject-bundled
make && make install
```

Vytvoření inicializačního skriptu, spuštění Asterisku a přihlášení do konzole, parametr -rvvvvv zajistí detailní výpis akci probíhající v Asterisku.

```
make config
/etc/init.d/asterisk start
root@debian:/home/user# asterisk -rvvvvv
```

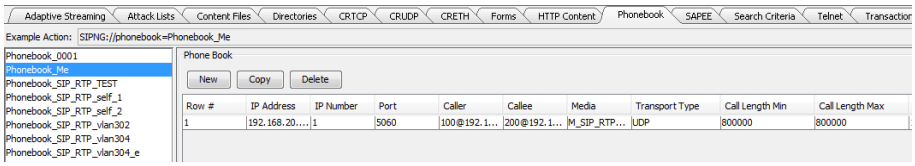
4.4 Konfigurace testeru

Testování scénářů je prováděno pomocí testeru Spirent TestCenter C1 Obr. 4.5. Jedná se komplexní tester síťové infrastruktury schopným generovat provoz na vrstvách L2-L3 do plné linkové rychlosti (1Gbps) k testování QoS (Quality of Service), na vrstvách L4-L7 do cca 5Gbps k testování aplikací a aplikačních protokolů. Tester podporuje celou řadu protokolů IPv4, IPv6, TCP, UDP, SSLv3,TLSv1, HTTP2, HTTPS, DNS, PPTP, Remote Desktop, MYSQL, SKYPE, SIP over TCP, SIP over UDP, a mnoho dalších. Seznam všech podporovaných protokolů je možné zjistit z [9].

Obr. 4.5: Spirent TestCenter C1

Pomocí grafického uživatelského programu Avalanche Commander je možné generovat provoz na čtyřech portech testeru. Kromě tohoto programu jsou k dispozici i podpůrné programy TestCenter Application a Results Analyzer. Program umožňuje zobrazovat detailní Real-time statistiky probíhajícího testu, na základě těchto statistik je možné měnit parametry testu bez jeho zastavení.

Před spuštěním samotného testu bylo třeba ověřit funkčnost testeru. V podpůrném programu je třeba nejprve vytvořit profil klienta a serveru (představující dvě koncové zařízení mezi kterými budou generované hovory). Na kartě nastavení klienta v záložce Actions je vytvořen tzv. *Phonebook*, kde se nastaví číslo klienta, adresa ústředny, port, adresa volaného, délka hovoru, a přenášena zvuková stopa (Media). V záložce Profiles pro generování provozu je vybrána aplikace s názvem SIPng.

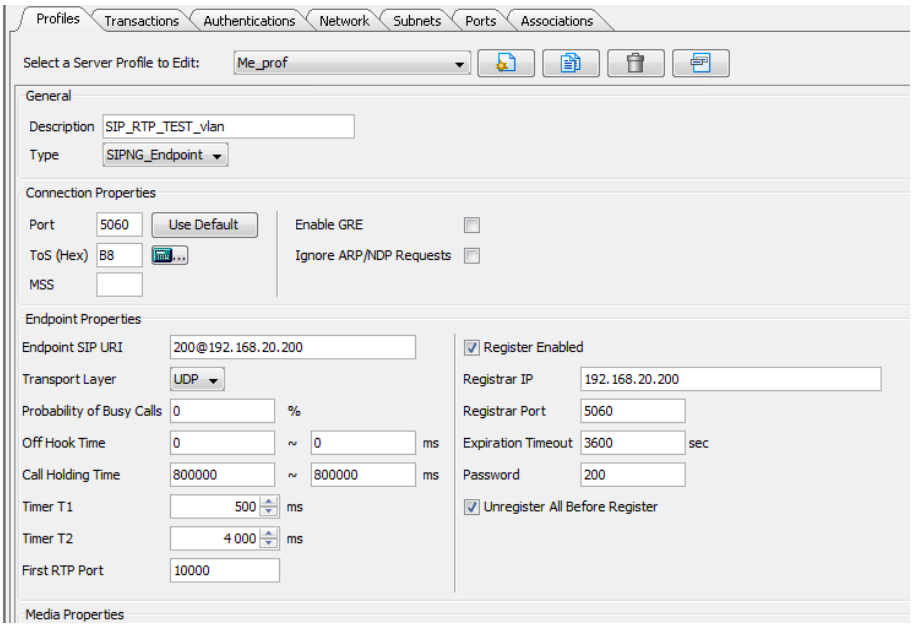


Obr. 4.6: Nastavení klienta

Na kartě serveru je nastaveno číslo klienta, adresa ústředny, heslo k registraci a délka hovoru. Obdobně zde byla nastavena i aplikace SIPng Obr. 4.7.

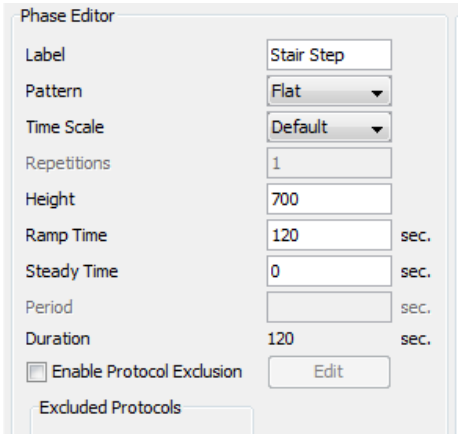
Následně je nutné nastavit příslušnou podsíť a asociovat vytvořené profily s porty. Po prvotním zkušebním testu, při kterém probíhali 4 hovory po dobu jedné minuty, bylo třeba v konfiguraci PJSIP vypnout podporu Session Timers, jelikož Asterisk hlásil chybu v SIP hlavičce. Při dalším testu na 100 hovorů bylo nutné zvýšit počet otevřených souborů Asteriskem, defaultní hodnota 1024 byla nedostatečná.

```
debian*CLI>ulimit descriptors 180839
```



Obr. 4.7: Nastavení serverové části

Po zkušebních testech na kartě klienta v záložce je nastaveno schéma generování hovorů podle obr. 4.12 viz. obr 4.8.



Obr. 4.8: Nastavení profilu Loads

4.5 Monitorování systému

Pro lepší vypovídající hodnotu výsledků testování byl použit skript, který zaznamenává aktuální vytížení procesoru do textového souboru s názvem cpu.txt, využití paměti RAM do souboru ram.txt a počet otevřených souborů Asteriskem do files.txt. Před každým spuštěním testu byla vyčištěna cache paměť a Asterisk byl restartován aby bylo zřejmé, kolik souborů ústředna vytváří při každém testu. Skript se spouštěl každou sekundu, toho bylo docíleno následujícím řádkem:

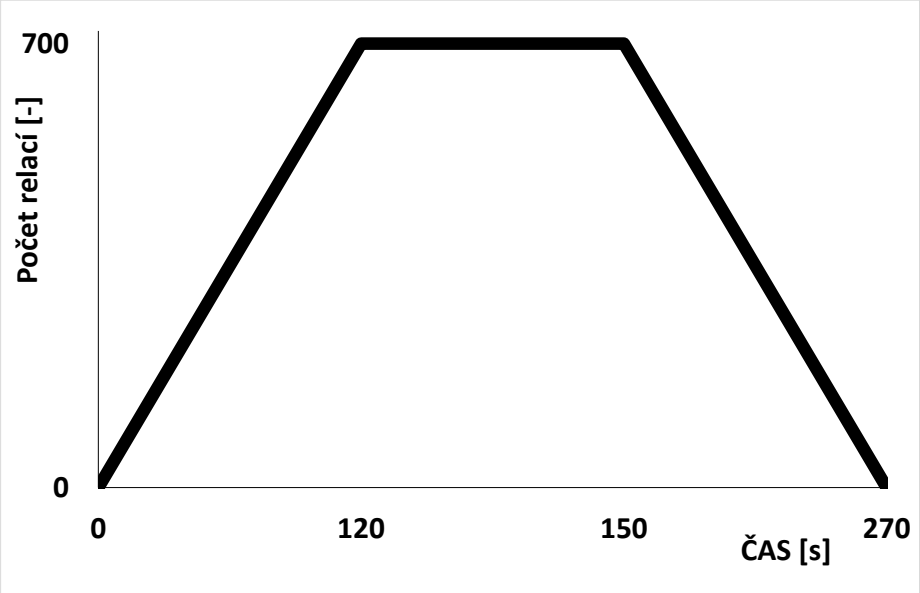
```
root@debian:homeuserPlocha# echo 3 >procsysvmdrop_caches
root@debian:/home/user/# watch -n 1 ./perf.sh
```

4.6 Testování maximálního počtu relací

Každý test byl prováděn třikrát, aby se minimalizovali nežádoucí vlivy na testování a výsledky tak měli lepší vypovídající hodnotu.

4.6.1 V B2BUA režimu

Nejprve byly provedeny zkušební testy využitím obou stacků, aby byla správně nastavena horní hranice maximálního počtu relací. Ta je nastavena o něco výše, než je ústředna schopna zpracovat. Ze zkušebních testů bylo odvozeno i schéma průběhu testování, které je znázorněno na obr. 4.12 V čase od 30 s začaly hovory postupně narůstat, až dosáhly maxima v čase 150 s, za 30 s byly postupně odpojovány. Po dokončení hovorů je zde ještě 30 s interval pro odpojení všech hovorů. Délka hovorů byla nastavena na 120+30s. Při testování v B2BUA režimu zřejmě nebudou výsledky obou stacků příliš rozdílné, protože ústředna musí zpracovávat RTP data klientů.



Obr. 4.9: Schéma testování

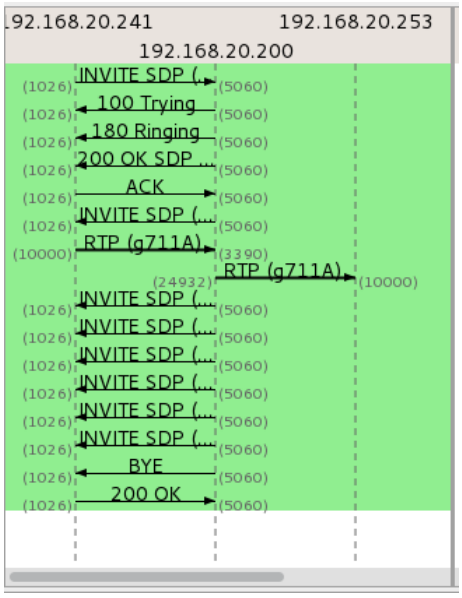
4.6.2 V Proxy režimu

V proxy režimu by ústředna měla dosáhnout lepších výsledků než v režimu B2BUA, z toho důvodu, že RTP data neprochází přes ústřednu a ústředna tak bude méně zatěžována. Byly provedeny zkušební testy z obdobných důvodů uvedených v kap. 4.6.1. Při využívání nativního stacku testy probíhali úspěšně. Při využívání PJSIP stacku se však vyskytl problém.

Při zasílání nové žádosti INVITE neboli re-INVITE po zahájení hovoru, tester Spirent tescenter C1 nedokázal na tento nový INVITE odpovědět, což vyústilo ze strany ústředny k následnému ukončení hovoru. Situaci znázorňuje obr. 4.10, kde IP 192.168.20.200 je adresa Asterisku a .253 a .241 jsou adresy testeru.

Jak je popsáno v kap. 3.5.1 nativní sip stack lze nastavit tak, aby neposílal re-INVITE s novými adresami v SDP, ale už v prvním zaslaném INVITE posílal v SDP adresu klienta a ne Asterisku, a tak při jeho testování tento problém nenastal. Jako první řešení tohoto problému se naskytla možnost zasílat místo žádosti INVITE žádost UPDATE, což však mělo za následek stejné chování jako v předešlém případě. Po prozkoumání všech možných i nemožných konfigurací nastavení stacku a testeru se nepovedlo chování upravit tak, aby tester na nový INVITE či UPDATE odpovídal.

Další možností bylo využít jiný tester. Konkrétně byl použit tester ABACUS



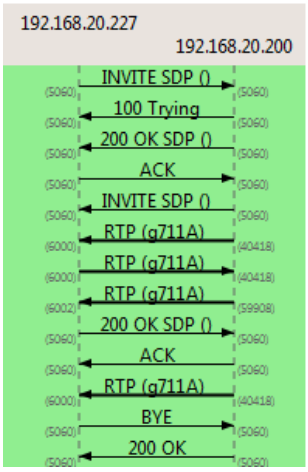
Obr. 4.10: Ukončení hovoru po neodpovězení na re-INVITE

5000, který nabízí oproti testeru Spirent C1, větší možnosti uprav chování SIP protokolu, ale nedokáže vytvořit tak velkou zátěž. Tento tester také představuje jak serverovou část, která hovory vytváří, tak i klientskou, která hovory přijímá. Po nastavení testeru popsaného např. v [13], byl proveden první test. Ze zachycené komunikace na obr. **??** bylo zjištěno, že tester již na re-INIVTE odpovídá korektně. Nicméně po potvrzení odpovědí tester stále zasílal RTP data na ústřednu a nikoliv na klienta. Z hlubší analýzy bylo zjištěno, že Asterisk posílá druhému klientovi v nové žádosti INVITE v SDP protokolu správnou multimediální adresu prvního klienta. Tester tuto zprávu potvrdí a odešle ve zprávě OK v SDP svou multimediální adresu, Asterisk dále obdobným způsobem přepošle tuto adresu druhému klientovi. Z čehož plyne, že tester má informace o tom kde se nachází jak server, tak i klient. Přesto stále zasílá RTP data na ústřednu.

Možnou příčinou mohlo být využívání stejného transportního portu pro multimédia na straně klienta i na straně serveru. Po změně portu u klienta však byla situace obdobná. Tester také nabízí možnost ovlivňovat odpovědi na příchozí žádosti. V tomto případě však odpovědi na re-INVITE či UPDATE byly korektní a tak nebylo nutné měnit způsob odpovídání na žádosti.

Nepovedlo se tedy tester přinutit ke změně adresy a portu kam má RTP data zasílat při obdržení nového INVITE či UPDATE. Tester se chová jako stavový automat a tak zasílá RTP data na adresu, kterou obdržel v prvním INVITE a na další změny adresy sice reaguje SIP zprávami, ale adresu odesílání nemění.

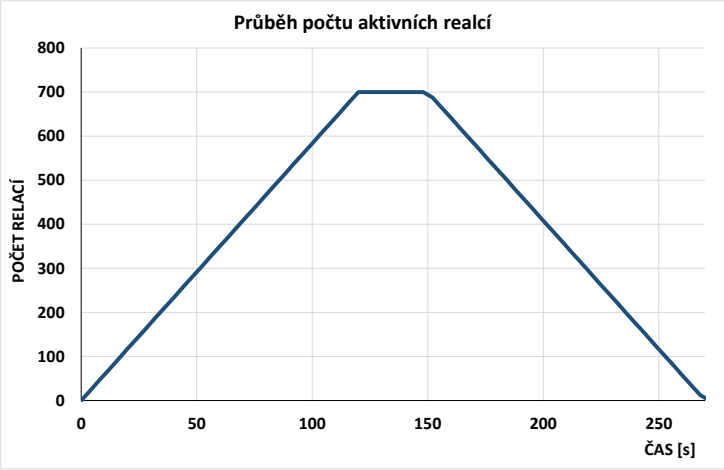
Z tohoto důvodu tak nebyl proveden test v proxy režimu.



Obr. 4.11: Kominikace při testování pomocí testeru Abacus

4.6.3 Výsledky testů na maximální počet relací

Zjištění maximálního počtu relací probíhalo tím způsobem, že se v grafu příchozích a odchozích RTP toků nalezl bod, kdy došlo k rozdělení těchto toků a následně byl z grafu aktivních relací zjištěn jejich maximální počet viz obr. 4.12. Při tomto rozdělení RTP toků začínalo taktéž narůstat procento zahozených RTP paketů, které se pohybovalo mezi 25% až 55 %. Rovněž se zvyšovala i odezva, ta se pohybovala okolo 800ms. Zhoršil se také jitter.



Obr. 4.12: Průběh aktivních relací

PJSIP stack

K rozdělení RTP dat docházelo 96 až 100 s, jak je znázorněno na Obr. 4.13. Asterisk s PJSIP stackem tak dosáhl 567 relací. Při zvyšování požadavků na další hovory již k většímu nárůstu relací nedošlo. Výkon procesoru se pohyboval mezi 60% - 70%. Zaplnění paměti vzrostlo v průměru o 212 MB. Počet otevřených souborů vzrostl z 32 na 9 880.

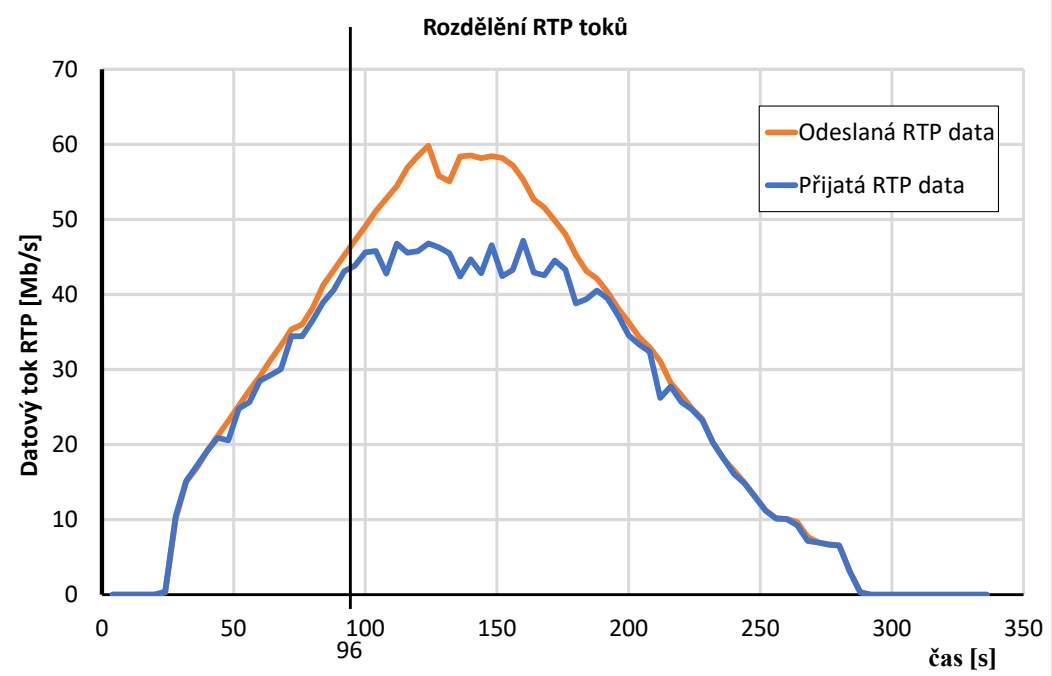
Obr. 4.13: Rozdělění RTP toků - PJSIP

	Měření 1	**Měření 2**	**Měření 3**	**Průměr**
Relace	560	583	559	**567**

Tab. 4.1: Maximální počet relací

SIP stack

Při implementaci SIP protokolu pomocí nativního stacku dosáhl Asterisk téměř identického počtu relací a to 575, kdy k rozdělení došlo ve stejný čas. Využití procesoru se pohybovalo mezi 70% - 80%. Využití paměti se zvýšilo o 234,25 MB. Počet vytvořených souborů byl 10 540.

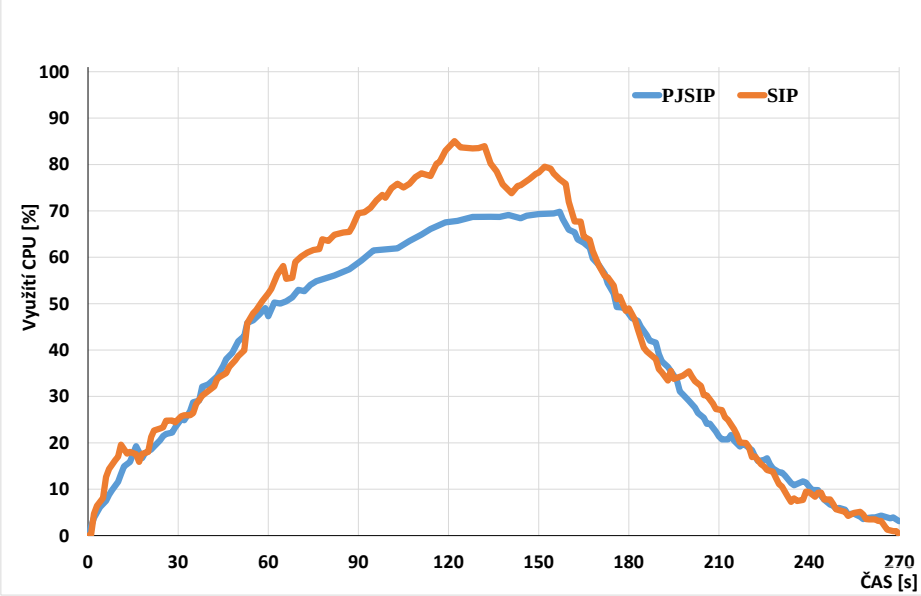


Obr. 4.14: Rozdělění RTP toků - SIP

	Měření 1	**Měření 2**	**Měření 3**	**Průměr**
Relace	583	582	560	**575**

Tab. 4.2: Maximální počet relací

Průběh využívání procesoru je na obr. 4.15. Pjsip stack využíval při plném zatížení ústředny procesor o cca 10% méně než nativní stack.



Obr. 4.15: Využití procesoru během testu

V následujících tabulkách je uvedena zaplnění paměti při každém ze 3 měření a stejně tak i počet vytvořených souborů:

	Měření 1	**Měření 2**	**Měření 3**	**Průměr**
PJSIP	1423,52	1419,52	1431,19	**1424,74**
SIP	1437,86	1420,19	1420,19	**1432,7**

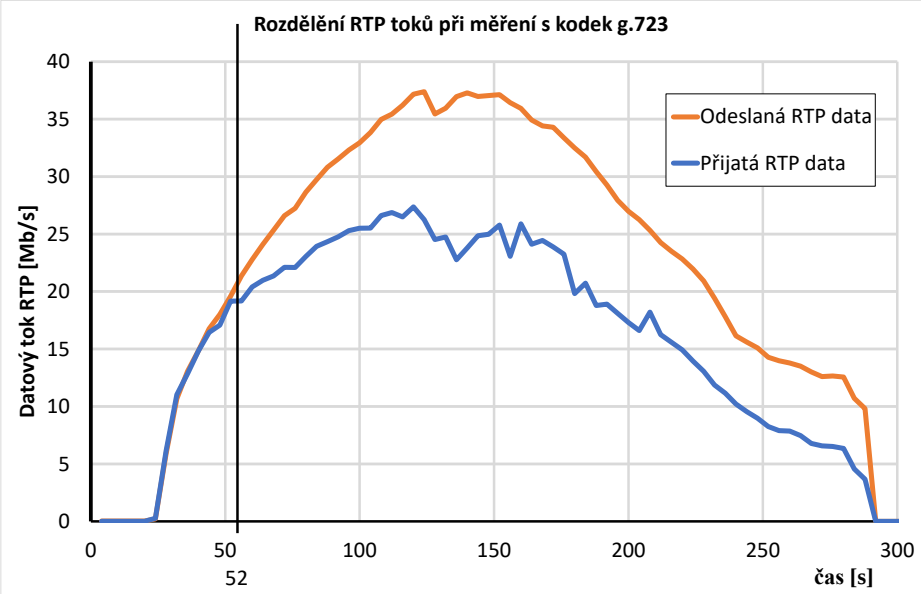
Tab. 4.3: Využití operační paměti

	Měření 1	Měření 2	Měření 3	Průměr
PJSIP	9832	9928	9879	**9880**
SIP	10482	10505	10643	**10540**

Tab. 4.4: Počet vytvořených souborů

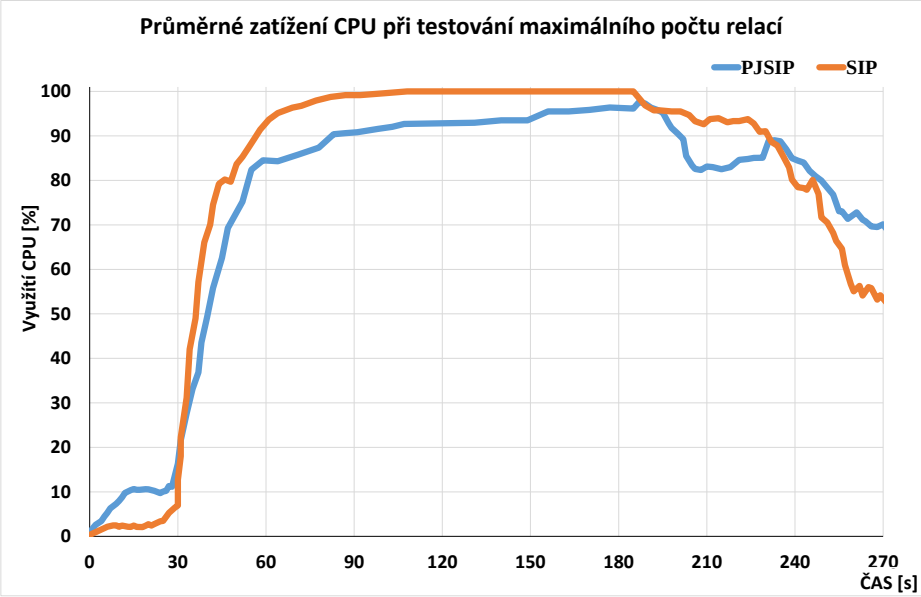
4.6.4 Měření relací s kodekem G.723

Limitujícím faktorem při měření relací tak bylo zpracování RTP dat. Proto proběhl ještě jeden obdobný test, při kterém byl snížen datový RTP tok. Toho bylo docíleno použitím jiného kodeku. Na místo kodeku a-law (G.711), který má bitový tok 64kb/s, byl použit kodek G.723, s bitovým tokem 6,3kb/s. Schéma průběhu testování bylo obdobné jako v kap. 4.6, maximální počet relací byl však nastaven na 2 000. S grafu 4.17 datových RTP toků je patrné snížení objemu RTP dat.



Obr. 4.16: Rozdělení RTP toků kodek g.723

Rozdělení RTP toků u obou stacku opět došlo zhruba ve stejný okamžik, a to okolo 52 vteřiny testování což odpovídá přibližně 867 relací. Vzrostlo však využití procesoru, nativní stack v době maximálního zatížení využíval procesor na 100%. PJSIP stacku taktéž vzrostlo využívání procesoru, to se pohybovalo okolo 95%.



Obr. 4.17: Využití procesoru stacky kodek g.723

Využití RAM bylo u obou stacku obdobné. PJSIP stack však vytvořil o téměř 2500 souborů méně než nativní stack.

	Čas rozdělení	**Počet relací**	**Využití RAM [GB]**	**Počet souborů**
PJSIP	52	867	1,433	17 420
SIP	52	864	1,437	19 912

Tab. 4.5: Počet vytvořených souborů

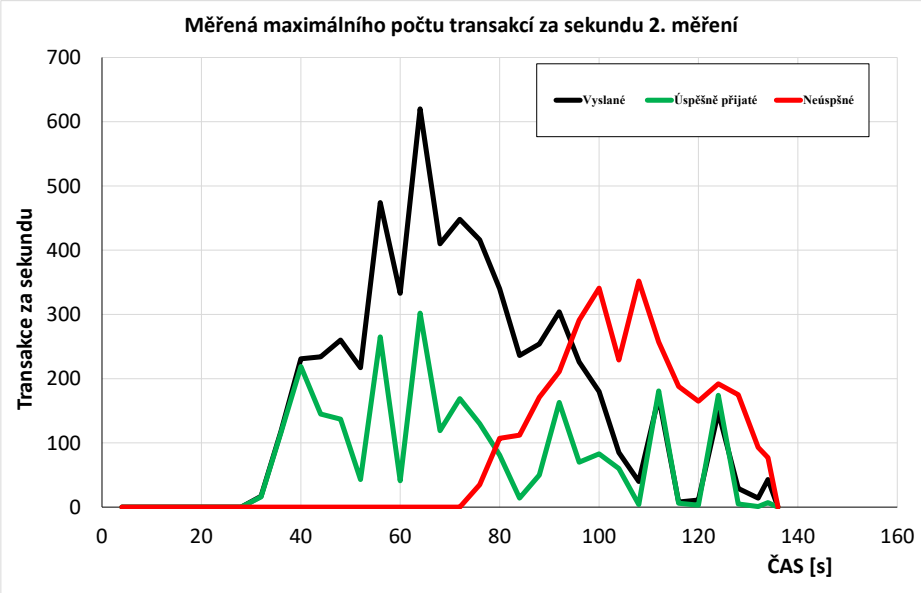
4.7 Měření maximálního počtu transakcí za sekundu

Cílem testu bylo zjistit maximální počet transakcí za sekundu, která je ústředna schopna zpracovat při využívání nativního a PJSIP stacku. Počet transakcí za sekundu postupně narůstal až na hodnotu 600 transakcí za sekundu. Při testech tester vyhodnocoval počet úspěšně a neúspěšně zpracovaných transakcí.

4.7.1 Výsledky měření maximálního počtu transakcí za sekundu

PJSIP stack

Maximálního počtu transakcí za sekundu ze všech testů dosáhl PSJIP při druhém testu v 68 s a to 302 transakcí za sekundu, průměrný počet transakcí ze tří testu byl 291. Při tomto testování byl procesor vytížen na 100%. Asterisk také vytvořil 17 209 souborů. Využití RAM bylo 1,422 GB.



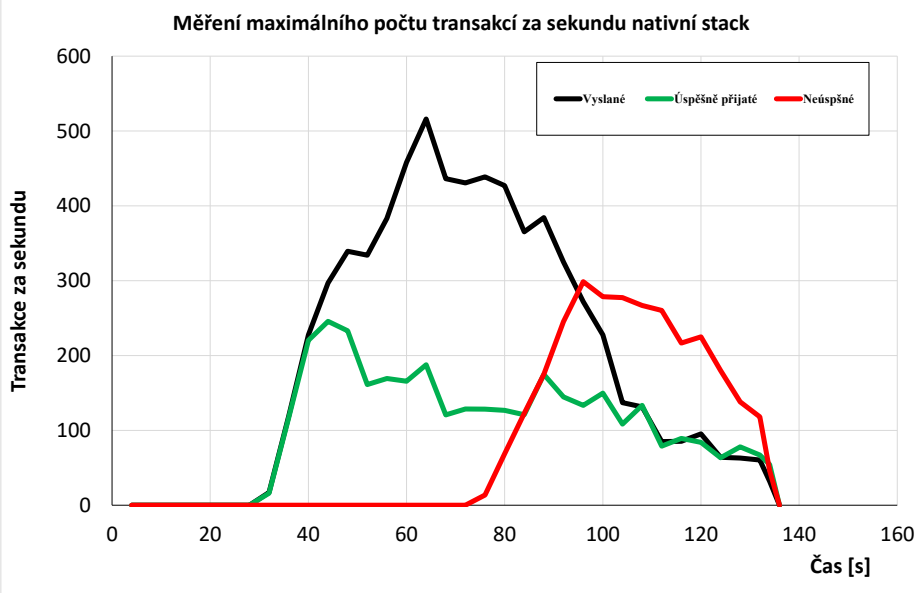
Obr. 4.18: Maximální počet transakcí za sekundu PJSIP stack

	Měření 1	**Měření 2**	**Měření 3**	**Průměr**
Počet transakcí	270	302	301	**291**

Tab. 4.6: Počet úspěšně přijatých transakcí za sekundu - PJSIP

SIP stack

Nativní SIP stack dosáhl o něco menšího počtu úspěšných transakcí. Maximální hodnota byla 295, v průměru to bylo v 271. Využití procesoru bylo opět 100%, souborů ovšem vytvořil 19 566, což je o cca 1 800 více než u PJSIP stacku, při téměř stejném počtu transakcí. Paměti RAM využil 1,436 GB, což je nepatrně více než u PJSIPu.



Obr. 4.19: Maximální počet transakcí za sekundu nativní stack

	Měření 1	**Měření 2**	**Měření 3**	**Průměr**
Počet transakcí	274	243	271	**291**

Tab. 4.7: Počet úspěšně přijatých transakcí za sekundu - SIP

4.8 Zhodnocení testů

Při testech na maximální počet relací dosáhli oba stacky téměř shodného počtu relací a to 567 při využití PJSIPu stacku resp. 575 při nativním stacku. Při sledování využití procesoru dosáhl nový PJSIP stack lepších výsledků, když zatížení procesoru bylo přibližně o 10% nižší než u nativního stacku viz. 4.15. Mírně lepších výsledků dosáhl nový stack i ve využívání operační paměti, kdy využíval 1,424 GB paměti oproti 1,432 GB u nativního stacku. Počet otevřených souborů se mezi stacky lišil o 660 souborů (PSJSIP 9880, nativní 10540). Při tomto testu bylo limitujícím faktorem zpracování RTP dat ústřednou, proto byl proveden další test, při kterém byl snížem objem RTP dat.

Při druhém testu na maximální počet relací byl použit místo kodeku a-law (G.711) kodek G.723. Při tomto testu oba stacky dosáhly přibližně o 300 úspěšných relací více než v předchozím. Rozdíl mezi stacky byl však co do počtu relací minimální. PJSIP stack dosáhl 867 relací a nativní 864 relací tab. 4.7. PJSIP stack využíval procesor přibližně o 5% méně než nativní stack viz obr. 4.17. Ve využívání RAM byly stacky opět téměř totožné. Nativní stack vytvořil o přibližně 2 500 více souborů než PJSIP stack. Při těchto testech docházelo k přetížení jádra Asterisku zpracováním RTP dat a tak implementace protokolu SIP neměla na počet aktivní relací výrazný vliv. Stacky se ovšem mírně liší ve využívání procesoru a počtem vytvářených souborů.

Při dalším měření byl zjišťován maximální počet transakcí za sekundu, které dokáže ústředna obsloužit. Při překročení hranice, kdy již ústředna není schopna zpracovávat více transakcí, dojde k *retransmissionům*. Limitujícím faktorem při tomto testu byl CPU, který při všech testech byl využíván na 100%. A tak obdobně jako v předešlých testech se obě implenetace SIP protokolu příliš neliší. PJSIP stack dosáhl 302 úspěšných transakcí, nativní stack dosáhl 295 úspěšných transakcí. Zaplnění operační paměti bylo opět mírně lepší u PJSIPu 1,422 GB, nativní 1,436 GB. I když dosáhly oba stacky téměř stejného počtu transakcí, vytvořil PJSIP stack v průměru o 2357 méně souborů než nativní stack.

Z dosažených výsledků testů je zřejmé, že mírně lepší výsledků dosáhla nová implementace SIP protokolu. Rozdíl je hlavně v hardwarové náročnosti stacků a to jak ve využívání CPU, RAM tak i počtu otevřených souborů.

4.9 Návrh laboratorní úlohy

V rámci diplomové práce je vytvořena laboratorní úloha do předmětu Telekomunikační a informační systémy. Tato laboratorní úloha má za cíl seznámit studenty s novou implementací SIP protokolu v ústředně Asterisk pomocí nového PJSIP stacku.

V úvodu laboratorní úlohy jsou studenti seznámeni s nový stackem, kde jsou vysvětleny důvody jeho zavedení a co nového přináší. Dále je zde ukázková konfigurace nastavení účtu pomocí PJSIP stacku. Následně je ukázáno jak lze vytvářet SIP trunk s využitím nového stacku a to dvěmi způsoby. Pomocí vytváření všech potřebných sekcí nutných k vytvoření SIP trunk a pomocí utility *Configuration Wizard* popsané v kapitole 4.1.3.

V praktické části mají studenti za úkol vytvoří 4 účty, registrovat zařízení na tyto účty a realizovat hovory mezi nimi. Poté ověří možnost registrace více zařízení na jeden účet a možnosti směrování hovorů na takový účet. V posledním úkole na ústředně nakonfigurují připojení k nadřazené ústředně pomocí trunků a následně upraví směrovací plán aby bylo možné přes tento trunk vytvářet a přijímat hovory z/do vnější sítě. Zadání laboratorní úlohy je uvedeno v příloze A.

5 ZÁVĚR

Cílem práce bylo porovnání nativního SIP stacku a PJSIP stacku sloužících k implementaci SIP protokolu v open source ústředně ASTERISK. V kapitole 2 je popsána a porovnána architektura stacků a jsou popsány jednotlivé moduly PJSIP stacku.

Byly vytvořeny čtyři scénáře uplatnění, kdy ústředna pracovala jako B2BUA (back-to-back user agent) v kap. 4.1.1, jako ústředna plnící funkci proxy severu kap.4.1.2, jako klient připojený k další ústředně kap. 4.1.3 a scénář kdy ústředna šifrovala spojení mezi ní a klienty kap. 4.1.4. K těmto scénářům byly vytvořeny komentované šablony konfigurace ústředny pomocí PJSIP stacku a pro porovnání i konfigurace pomoci nativního stacku. Tyto šablony jsou připojeny jako elektronické přílohy k diplomové práci.

Při testech maximálního počtu aktivních relací, kdy pracovala ústředna v režimu B2BUA, dosáhly oba stack téměř stejného počtu aktivních relací. PJSIP stack však dosáhl lepších výsledků při porovnání využívání procesoru. Výsledky se lišili až o 10% viz obr. 4.15. Nepatrně lepší byl i při využívání paměti RAM a počtu vytvořených souborů při testování. V B2BUA režimu docházelo k přetížení jádra ústředny a tak rozdíly implementace SIP protokolu neměli na počet relací velký vliv.

Dalším testovaným parametrem byla schopnost zpracovávat velké množství transakcí za sekundu. V tomto parametru dosáhly opět oba stacky téměř shodných výsledků. PJSIP stack vytvořil méně souboru při stejném počtu úspěšně zpracovaných transakcí.

Přínos nově nasazovaného stacku je v jeho architektuře. Jak je zmíněno v kap. 2.2, architektura je vysoce modulární a je tak lépe připravená na úpravu či přidání nových funkcí spojených s vylepšováním SIP protokolu. PJSIP stack je méně náročný na využívání hardware než původní stack. Nový stack nabízí také komplexnější možnost konfigurace SIP protokolu pomocí logických sekcí.

V rámci diplomové práce bylo vytvořeno zadání laboratorní úlohy do předmětu Telekomunikační a informační systémy sloužící k seznámení s PJSIP stackem a jeho základní konfigurací. Zadání je připojeno jako příloha A.

LITERATURA

[1] ČERVENKA, Marek. *Asterisk 12 – New Age* ROOT.cz [online]. 2014, , 1 [cit. 2016-12-10]. Dostupné z: <https://www.root.cz/clanky/asterisk-12-new-age/>

[2] CHALÁS, JAROSLAV. *Metody zajištění bezpečnosti VoIP provozu Open source PBX* [online]. Brno, 2010 [cit. 2016-10-22]. Dostupné z: <https://www.vutbr.cz/studium/zaverecne-prace?zp_id=26636>

[3] P. HAGENDORF. *XML Schema for Media Control*: RFC 5168 [online]. 2008 [cit. 2016-12-2]. Dostupné z: <https://tools.ietf.org/html/rfc5168>

[4] ŠILHAVÝ, PH.D, Ing. Pavel. *Telekomunikační a informační systémy.* BRNO, 2014. Vysoké učení technické v Brně Fakulta elektrotechniky a komunikačních technologií Ústav telekomunikací Technická 12, 616 00 Brno.

[5] VOZŇÁK, Miroslav. *Technologie a protokoly multimediálních komunikací pro integrovanou výuku VUT a VŠB-TUO.* Ostrava: Vysoká škola báňská - Technická univerzita Ostrava, 2014. ISBN 978-80-248-3326-2.

[6] The Session Initiation Protocol - The Internet Protocol Journal - Volume 6, Number 1. CISCO [online]. [cit. 2016-12-10]. Dostupné z: <http://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-23/sip.html>

[7] STELIOS ANTONIOU. *VoIP Signaling Protocols:* RFC 5168 [online]. 2010 [cit. 2016-12-2]. Dostupné z: <https://www.pluralsight.com/blog/it-ops/voip-signaling-protocols>

[8] Asterisk Versions. *Asterisk.org* [online]. Russell Bryant, 2016 [cit. 2016-10-20]. Dostupné z: <https://wiki.asterisk.org/wiki/display/AST/Asterisk+Versions>

[9] Spirent Avalanche [online]. 2013, poslední aktualizace duben 2013 [cit. 18. 11. 2016]. Security testing and application performance. Dostupné z URL: <http://www.spirent.com/~/media/Datasheets/Broadband/PAB/Avalanche/Avalanche_Security_Testing_and_Application_Performance_Datasheet.pdf>

[10] DAVENPORT, Malcolm. *Secure Calling Tutorial.* WIKI ASTERISK [online]. 2016 [cit. 2017-03-20]. Dostupné z: <https://wiki.asterisk.org/wiki/display/AST/Secure+Calling+Tutorial>

[11] Mirror of the official Asterisk Project repository. GitHub [online]. [cit. 2017-04-03]. Dostupné z: <https://github.com/asterisk/asterisk>

[12] YouTube video Kamailio World 2014 - Matt Jordan - *Asterisk 12 and the PjSIP Channel In: Youtube* [online]. 3. 7. 2014 [cit. 2017-04-11]. Dostupné z: <https://www.youtube.com/watch?v=N010yBQmQJ0> Kanál uživatele Kamailio World

[13] BEDNÁR, Jakub Výkonnostní limity, spolehlivost a bezpečnost Open source PBX: diplomová práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2012. 145 s. Vedoucí práce bol Ing. Pavel Šilhavý, Ph.D.

[14] SIP Direct Media Reinvite Glare Avoidance. *Wiki Asterisk* [online]. 2012 [cit. 2017-05-17]. Dostupné z: <https://wiki.asterisk.org/wiki/display/AST/SIP+Direct+Media+Reinvite+Glare+Avoidance>

[15] JOHNSTON, Alan B. *SIP: understanding the Session Initiation Protocol.* 3rd ed. Boston: Artech House, c2009. ISBN 9781607839958.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

B2BUA Back-to-Back user adent

CLI Command-line Interface

DNS Domain Name Server

DTLS Datagram Transport Layer Security

DTMF Dual-tone multi-frequency

IAX2 Inter-Asterisk eXchange

IP Internet protocol

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

IVR Interactive voice response

LTS Long Term Support –verze s dlouhodobou podporou

MGCP Media Gateways Control Protocols

MWI Message Waiting Indicator

NAPTR Name Authority Pointer

PBX Private branch Exchange

PC Personal Computer

PSTN Public Switched Telephone Network

RFC Request For Comments

RTP Real Time Protocol – protokol pro přenos multimediálních dat

SDP Session Description Protocol

SIP Session Initiation Protocol

SRTP Secure Real-time Transport Protocol

TCP Transmission Control Protocol

TDM Time Division Multiplex

TLS Transport Layer Security

UAC User Agent Client

UAS User Agent Server

UDP User Datagram Protocol

URI Uniform Resource Identifier

VoIP Voice over IP

XML Extensible Markup Language

SEZNAM PŘÍLOH

A ZADANÍ LABORATORNÍ ÚLOHY

5 PBX Asterisk – PJSIP stack

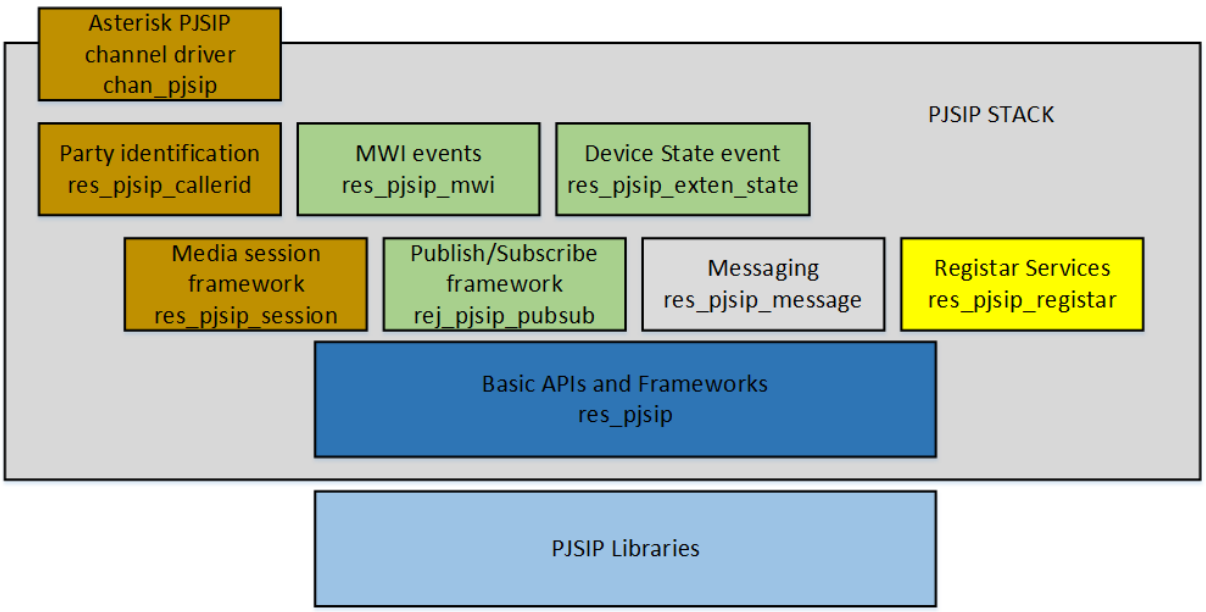
5.1 Úvod

V minulých cvičení jste se seznámili s konfigurací SIP kanálu Asterisku pomocí nativního kanálového ovladače. V tomto cvičení se seznámíte s konfigurací SIP kanálu pomocí nového ovladače PJSIP.

5.2 PJSIP stack

V PBX Asterisk se o implementaci každého protokolu starají tzv. kanálové ovladače neboli stacky. Pro protokol SIP jsou zde dva stacky původní s názvem res_sip a nový stack s názvem res_pjsip. Původní stack byl navržen již v roce 2002. S postupným vývojem SIP protokolu bylo však stále náročnější upravovat tento původní stack. A tak je v Asterisku od verze 12 implementován vedle původního stacku i nový PJSIP stack. Název pochází z knihoven PJSIP, na kterých je založený. Při návrhu nového stacku byla snaha zachovat co nejlepší funkčnost shodnou s původním stackem. PJSIP stack však přináší i některé nové funkce například možnost registrace více zařízení na jeden účet či podporu protokolu HEP (Homer Encapsulation Protocol).

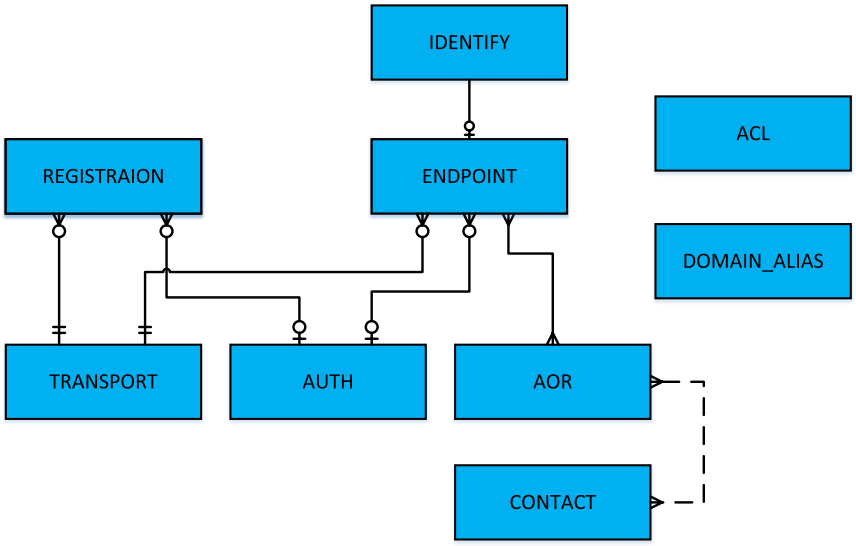
Hlavní rozdíl je však v architektuře, stack je navržen jako vysoce modulární, kde každý modul plní jinou funkci. Například modul s názvem res_pjsip_registar zajišťuje registrace atd. Díky této architektuře je snazší přidávání nových funkcí či upravování těch stávajících bez nutnosti zásahu do jádra stacku. Architektura nového stacku je znázorněna na Obr. 1 .



Obr. 1 Architektura PJSIP stacku

5.2.1 Konfigurace

PJSIP stack má odlišnou konfiguraci oproti nativnímu stacku. Konfigurace účtů je prováděna pomocí logických sekcí, kde každá sekce upravuje chování nějakého modulu ve stacku. Na Obr. 2 jsou znázorněny sekce nastavení a vazby mezi nimi.



Obr. 2 Vazby mezi sekcemi nastavení

ENDPOINT

Slouží pro konfiguraci koncového bodu (účtu), např. se zde definuje kontext, používané kodeky a přidružení konkrétních sekcí dalších typů. Jedna sekce typu ENDPOINT může být přidružena k jedné sekci AUTH, TRANSPORT, AOR a INDENTIFY

TRANSPORT

Určuje chování na transportní vrstvě. Jedna sekce transport může být přidružena k více sekcím ENDPOINT i REGISTRATION.

AUTH

Určuje způsob ověřování při registraci klientů i Asterisku jako klienta. Tato sekce může být přidružena k více ENDPOINT i REGISTRATION.

AOR

Cílem sekce AOR je sdělit Asterisku, kde má kontaktovat ENDPOINT. Bez této sekce nelze ENDPOINT kontaktovat. AOR sekce může být přidružena k více ENDPOINT a CONTACT

REGISTRATION

Slouží k nastavení pro odchozí registraci (SIP trunk). Jedna REGISTRATION sekce může být přidružena k jedné sekci TRANSPORT a AUTH.

DOMAIN_ALIAS

Pokud přichází provoz z domény, pro kterou není v AOR shoda, je možné zvolit pro

tuto doménu tzv. alias. Nemá přímý vztah s jinou sekcí.

ACL

Ovlivňuje veškerou příchozí komunikaci přes modul res_pjsip a je nezávislá na koncovém bodu. Nemá přímý vztah s jinou sekcí.

IDENTIFY

Slouží k identifikaci, k jakému koncovému bodu patří příchozí paket. Jedna sekce IDENTIFY může být přidružena k sekci ENDPOINT.

CONTACT

Slouží k vytvoření SIP

Například pro konfiguraci běžného účtu je nutné nejdříve vytvořit sekci typu ENDPOINT, kde jsou definovány základní vlastnosti účtu (kodek, kontext). A k této sekci přiřadit další sekce, sekce typu TRANSPORT, AOR a AUTH.

Konfigurace se provádí v textovém souboru pjsip.conf, který se nachází ve stejném adresáři jako ostatní konfigurační soubory ústředny (sip.conf, extensions.conf). Níže je uveden příklad konfigurace pro účet 500.

```
[transport_udp]                 ;název sekce
type=transport                  ;typ sekce
protocol=udp                    ;protokol
bind=0.0.0.0:5064               ;adresa, port na které Asterisk naslouchá

[500]                           ;název sekce
type=endpoint                   ;typ sekce
context=kontext1                ;kontext do kterého učet patří
disallow=all                    ;zakáže všechny kodeky
allow=alaw                      ;povolení kodeku alaw
transport=transport_udp         ;přiřazení sekce transport
auth=auth500                    ;přiřazení sekce AUTH
aors=500                        ;přiřazení sekce AOR


[auth500]                       ;název sekce
type=auth                       ;typ sekce
auth_type=userpass              ;typ ověřování
username=500                    ;uživ. jméno
password=asdf                   ;heslo

[500]                           ;název sekce
type=aor                        ;typ sekce
max_contacts=1                  ;maximální počet připojených zařízení
```

Vytváření SIP trunku

Vytváření trunků v PJSIP stacku v Asterisku od verze 12 usnadňuje utilita *PJSIP Configuration Wizard*, která zajistí vytvoření všech potřebných sekcí nutných k úspěšnému vytvoření trunku. Konfigurace utility se provádí v textovém souboru pjsip_wizard.conf. Pro vytvoření trunků tedy postačí vyplnit položky, jak je uvedeno níže. Poté stačí reloadovat pjsip stack.

```
[SIP_TRN_1]                         ;název
type=wizard                         ;typ sekce
sends_auth=yes                      ;použití jména a heslo pro reg
sends_registrations=yes             ;registrovat se k ústředně
remote_hosts=192.168.10.5           ;adresa ústředny kam se máme registrovat
outbound_auth/username=SIP_TRN_0    ;uživatelské jméno
outbound_auth/password=anaca        ;heslo
endpoint/context=prichozí_kontext ;context
endpoint/allow=alaw                 ;povoleni kodeku alaw
```

Pro vytvoření trunků je také možné kromě této utility, nastavit všechny potřebné sekce ručně v konfiguračním souboru pjsip.conf. Je nutné vytvořit sekci REGISTRTION, která zajistí registraci u nadřazené ústředny pomocí uživ. jména a hesla definovaného v sekce ATUH. Sekce ENDPOINT představuje účet, na který budou směrovány odchozí hovory přes vytvořený trunk. Kde se má tato ústředna kontaktovat určuje položka contact v sekci AOR. Sekce IDENTIFY identifikuje příchozí provoz z IP adresy nadřazené ústředny jako ENDPOINT SIP_TRN_1.

```
[SIP_TRN_1]                                 ;název
type=endpoint                               ;typ sekce
aors= SIP_TRN_0                             ;přiřazení sekce AOR
outbound_auth=SIP_TRN_0_auth                ;přiřazení sekce AUTH
context= prichozí_kontext                   ;context

[SIP_TRN_1]                                 ;název
type=aor                                    ;typ sekce
contac=192.168.10.5                         ;kontaktní adresa

[SIP_TRN_0_auth]                            ;název
type=auth                                   ;typ sekce
auth_type=userpass                          ;typ ověřování
username= SIP_TRN_1                         ;uživ. Jméno
password=heslo                              ;heslo

[SIP_TRN_1-reg]                             ;název sekce
type=registration                           ;typ sekce
outbound_auth= SIP_TRN_1_auth               ;použití sekce auth
server_uri= sip:192.168.10.5                ;URI adresa serveru
client_uri=sip:SIP_TRN_0@192.168.10.5       ;klientská URI

[SIP_TRN_1-identify]                        ;název sekce
type=identify                               ;typ sekce
endpoint= SIP_TRN_1                         ;přiřazení sekce endpoint
match= 192.168.0.5                          ;identifokování podle adresy
```

5.3 Zadání úlohy – PJSIP stack

Úkol 1: Seznamte se se způsobem konfigurace PJSIP stacku dle kapitoly 5.2.

Úkol 2: Vytvořte účty vyžívajících PJSIP

Podle vzoru v kapitole 5.2 vytvořte v souboru pjsip.conf další účty (sekce ENDPOINT) s názvy 1500 1600, 1700, a k nim potřebné sekce(AUTH, AOR). Sekce typu AOR musí mít shodný název se sekcí ENDPOINT, ke které je přiřazena. Sekce Auth může mít libovolný název, v sekci ENDPONIT však musíme tuto sekci správně přiřadit. Sekci typu TRANSPORT můžete ponechat pouze jednu *transport_udp* a přiřadit ji i k ostatním účtům. Po reloadování stacku příkazem *pjsip relaod*, ověřte správnost konfigurace příkazem show *pjsip endpoints* Obr. 3.

```
==========================================================================================

Endpoint:  1500                                                 Unavailable   0 of inf
    InAuth:  auth1500/1500
       Aor:  1500                                               2
 Transport:  transport_udp            udp      0      0  0.0.0.0:6060
```

Obr. 3 Výpis konfigurace ENDPOINT

Úkol 3: Upravte směrovací pravidla, aby Asterisk využíval PJSIP stack

Po ověření správnosti konfigurace upravte směrovací pravidla v kontextu *kontext3* tak aby hovory na klapky 1500-1700 využívali stack PJSIP. K volání klapky PJSIP stackem místo nativním stackem lze dosáhnou obdobným způsobem jako při využívání SIP a IAX2 protokolu:

```
[kontext3]
exten =>500,1,Dial(PJSIP/500)
```

Dále zaregistrujte telefony na vytvořené účty. Nezapomeňte změnit port z hodnoty 5060 na 5064 definovaném v sekci TRANSPORT. Ověření správné registrace se opět provádí příkazem *show pjsip endpoints* Obr. 4

```
==========================================================================================

Endpoint:  1500                                                 Not in use    0 of inf
    InAuth:  auth1500/1500
       Aor:  1500                                               2
   Contact:  1500/sip:1500@192.168.20.120:5060;transpor f0a0f4966f Unknown         nan
 Transport:  transport_udp            udp      0      0  0.0.0.0:6060
```

Obr. 4 Registrované zařízení k ENDPOINT

Úkol 4: Zaregistrujte více zařízení na jeden účet

PJSIP stack umožňuje registraci více zařízení na jeden účet. Například softwarový klient a hardwarový telefon. Maximální dovolený počet připojených zařízení se definuje v sekci typu AOR položkou *max_contacts.* Na takovouto klapku je možné provést hovor na první přihlášené zařízení:

```
[kontext3]
exten =>1500,1,Dial(PJSIP/1500)
```

Nebo na všechny registrované zařízení:

```
[kontext3]
exten =>1500,1,Dial(${PJSIP_DIAL_CONTACTS(${EXTEN})})
```

Upravte nastavení účtu 1500 tak, aby bylo možné na ni registrovat více zařízení. Zaregistrujte na tuto klapku dvě zařízení například hardwarový telefon a softwarového klienta v PC. Při úspěšné registraci obou zařízení jsou při výpisu informací o endpointu 1500 vidět dvě adresy těchto zařízení u položek *contact* Obr. 5.

```
==========================================================================================

Endpoint:  500                                           Not in use    0 of inf
    InAuth:  auth500/500
       Aor:  500                                         2
     Contact:  500/sip:500@192.168.20.200:5061;transport= 46771a9b06 Unknown         nan
     Contact:  500/sip:500@192.168.20.120:5060;transport= aa45ed3651 Unknown         nan
 Transport:  transport_udp            udp      0      0  0.0.0.0:6060
```

Obr. 5 Dvě registrovaná zařízení na jeden účet

Následně zavolejte klapku a ověřte, že je hovor směrován na první připojený kontakt. Poté upravte směrovací pravidla, aby při volání této klapky vyzváněli všechny telefony registrované na tuto klapku.

🛠 Úkol 5: Vytvořte SIP trunk k ústředně Anča (192.168.10.5) s využitím PJSIP stacku

Nejprve zakomentujte v souboru sip.conf trunk vytvořený v laboratorní úloze č. 3 a relaodujte sip stack příkazem *sip reload.*

Vytvořte SIP trunk mezi vaší ústřednou a serverem Anča (192.168.10.5) podle schématu kap. 5.2.1. Přidělení trunků k pracovištím je stejné jako v úloze č. 3 viz. **Tab. 1**

Tab. 1: Rozdělení pracovišť

Pracoviště	IP adresa PC	IP adresa telefonu	klapky	Jméno trunku – IP 192.168.10.5
1	192.168.10.101	192.168.10.161	60X	SIP_TRN_0 nebo IAX2_TRN_0
2	192.168.10.102	192.168.10.162	61X	SIP_TRN_1 nebo IAX2_TRN_1
3	192.168.10.103	192.168.10.163	62X	SIP_TRN_2 nebo IAX2_TRN_2
4	192.168.10.104	192.168.10.164	63X	SIP_TRN_3 nebo IAX2_TRN_3
5	192.168.10.105	192.168.10.165	64X	SIP_TRN_4 nebo IAX2_TRN_4
6	192.168.10.106	192.168.10.166	65X	SIP_TRN_5 nebo IAX2_TRN_5
7	192.168.10.107	192.168.10.167	66X	SIP_TRN_6 nebo IAX2_TRN_6
8	192.168.10.108	192.168.10.168	67X	SIP_TRN_7 nebo IAX2_TRN_7
9	192.168.10.109	192.168.10.169	68X	SIP_TRN_8 nebo IAX2_TRN_8
10	192.168.10.110	192.168.10.170	69X	SIP_TRN_9 nebo IAX2_TRN_9

Pozn.: Heslo je pro všechny trunky **anca**.

Opět restartujte PJSIP stack a ověřte úspěšnou registraci k Anče příkazem *pjsip show registrations.*

```
PC425*CLI> pjsip show registrations

 <Registration/ServerURI..............................>  <Auth.........>  <Status.......>
==========================================================================================

 SIP_TRN_0-reg-0/sip:192.168.10.5                        SIP_TRN_0-oauth   Registered
```

🛠 Úkol 6: Upravte směrovací pravidla, aby bylo možné volat do vnější sítě přes 0 s využitím PJSIP stacku.

Upravte směrovací pravidla v kontextu *kontext3*, obdobně jako v laboratorní úloze č.3 úkol 5. |Upravte tedy identifikaci volajícího CallerID na číslo v rámci veřejného číslovacího plánu, tedy přepsání čísla ústředny (např. 1100) na veřejné číslo volající ústředny (např. 611) pomocí funkce Set. Dále pravidla upravte, tak aby bylo možné dovolat se přes 0 na ostatní pracoviště s využitím SIP trunku vytvořeného PJSIP stackem. Níže je uveden příklad pro pracoviště 2:

```
exten =>_0XXX,1,Set(CALLERID(num)=61${CALLERID(num):1:1})
exten =>_0XXX,2,Dial(PJSIP/${EXTEN:1}@SIP_TRN_1)
```

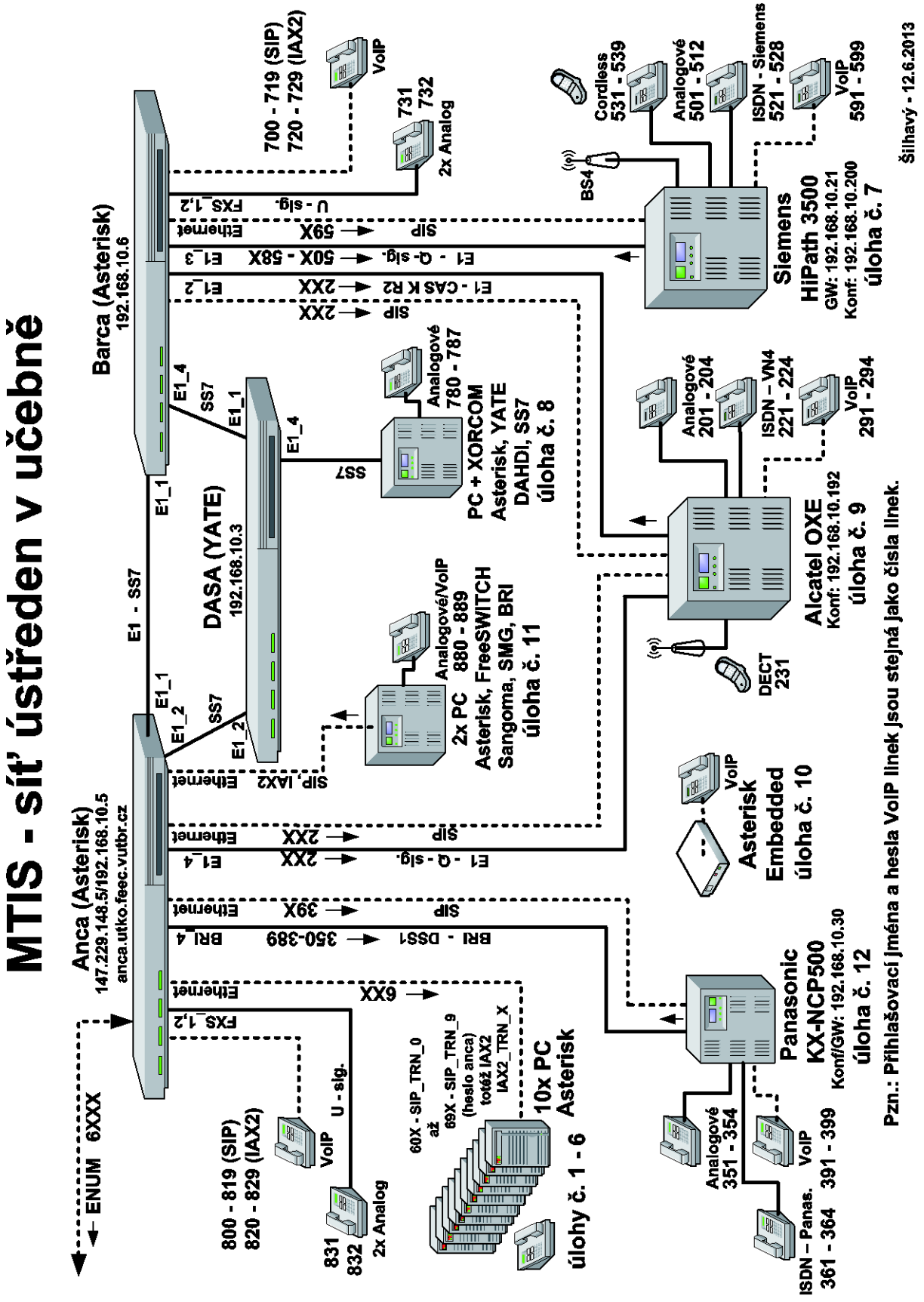
Funkčnost uvedeného zápisu volání na 6xx budeme moci ověřit až kolegové realizují konfiguraci dle úkolu č. 7 nebo voláním na klapky 810-810 (DECT).

🛠 Úkol 7: Upravte směrovací pravidla, aby byla příchozí volání směrována na interní klapky.

Nyní je možné dovolat se z interní sítě do vnější. Aby bylo možné dovolat se i z vnější sítě do interní je třeba upravit kontext *prichozi_kontext*, který určuje, jaká klapka bude volána pokud přijde hovor na vnější čísla pracoviště (u pracoviště 2 čísla 61x). Upravte směrovací pravidla, aby hovory na číslo 6x5 byly směrovány na 1500, na číslo 6x6->1600 a na číslo 6x7->1700 (účty yvtvořené PJSIP stackem) obdobným způsobem jako v lab. 3 úkol 7.

```
[prichozi_kontext]
exten =>_60[12],1,Dial(SIP/1${EXTEN:2:1}00)
exten =>_60[34],1,Dial(IAX2/2${EXTEN:2:1}00)
exten =>_60[567],1,Dial(PJSIP/1${EXTEN:2:1}00)
exten =>_60[89],1,Dial(dahdi/1)
```

5.4 Příloha 1: Mapa ústředen v laboratoři



Seznam použité literatury

[1] ŠILHAVÝ, PH.D, Ing. Pavel. *Telekomunikační a informační systémy*. BRNO, 2014. Vysoké učení technické v Brně Fakulta elektrotechniky a komunikačních technologií Ústav telekomunikací Technická 12, 616 00 Brno.

[2] YouTube video Kamailio World 2014 - Matt Jordan - Asterisk 12 and the PjSIP Channel In: Youtube [online]. 3. 7. 2014 [cit. 2017-04-11]. Dostupné z: https://www.youtube.com/watch?v=N010yBQmQJ0 Kanál uživatele KamailioWorld

[3] Asterisk 12 Part IV: *The SIP Stack of the Future:* Matt Jordan. Blogs.digium.com [online]. [cit. 2016-11-8]. Dostupné z: http://blogs.digium.com/2013/11/20/asterisk-12-part-iv-sip-stack-future/

[4] *New in 13*: Matt Jordan. Wiki.asterisk [online]. 2014 [cit. 2016-11-8]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/New+in+13

[5] *Asterisk Versions*. Asterisk.org [online]. Russell Bryant, 2016 [cit. 2016-10-20]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/Asterisk+Versions

[6] SIP: Session Initiation Protocol, Internet Engineering Task Force, [online] http://www.ietf.org/rfc/rfc3261.txt

[7] RTP: Real-time Transport Protocol, Javvin, [online] http://www.javvin.com/protocolRTP.html

[8] PJSIP Configuration Wizard. *Asterisk wiki* [online]. 2016 [cit. 2017-05-15]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/PJSIP+Configuration+Wizard

[9] Dialing PJSIP Channels. *Asterisk wiki* [online]. 2014 [cit. 2017-05-15]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/Dialing+PJSIP+Channels

[10] PJSIP Configuration Sections and Relationships. *Asterisk wiki* [online]. 2016 [cit. 2017-05-15]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/PJSIP+Configuration+Sections+and+Relationships

B OBSAH PŘILOŽENÉHO CD

Na přiloženém CD se nacházejí soubory:

- DP_xbedna54.pdf - elektronická verze této práce
- složka Namerena_data - obsahující data a grafy změřené při testech v kapitolách 4.6.3,4.6.4, 4.7.1
- složka Sablony - obsahující komentované konfigurační soubory ústředny pro scénáře popsané v kapitole 4.1
- složka Lab_uloha - obsahující zadání laboratorní úlohy a předpřipravené konfigurační soubory

C SKRIPT PRO MONITOROVÁNÍ SYSTÉMU

```
# !/bin/bash

# RAM
date '+%X' | tr '\n' ' ' » ram.txt
free | grep Mem | awk '{print ($2-$4)/1024}' » ram.txt

# CPU
date '+%X' | tr '\n' ' ' » cpu.txt
cat <(grep 'cpu ' /proc/stat) <(sleep 0.1 && grep 'cpu '
/proc/stat) | awk -v RS="{print ($13-$2+$15-$4)*100/
($13-$2+$15-$4+$16-$5)}' » cpu.txt

# otevřené soubory
date '+%X' | tr '\n' ' ' » files.txt
ls -l '/proc/'$(pidof asterisk)'/fd'| wc -l » files.txt
```